The Thesis Committee for Aprajita Sant
Certifies that this is the approved version of the following thesis

SCREENING PROCEDURE TO IDENTIFY POWER SYSTEM EVENTS OF THE TEXAS SYNCHROPHASOR NETWORK

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

W. Mack Grady

Surya Santoso

SCREENING PROCEDURE TO IDENTIFY POWER SYSTEM EVENTS OF THE TEXAS SYNCHROPHASOR NETWORK

by

Aprajita Sant, B.Tech

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

May 2012

Dedication

I dedicate this work to my grandfather Mr. H.S. Sant and my parents Dr. Anjali Sant and Mr. Ajit Sant and my younger sister Yesha Sant who have been a source of inspiration to me and encouraged me to fulfill my dreams.

Acknowledgements

I am very grateful to Dr. Grady for his constant support, encouragement and valuable guidance that helped me complete my studies successfully. I would like to thank Dr. Santoso for his support and guidance. I am thankful to my friends Sanujit and Anamika for their help.

ABSTRACT

SCREENING PROCEDURE TO IDENTIFY POWER SYSTEM EVENTS OF THE TEXAS SYNCHROPHASOR NETWORK

Aprajita Sant, MSE

The University of Texas at Austin, 2012

Supervisor: W. Mack Grady

This work presents a method for screening synchrophasor data to search for power system events of interest. The method employs prony algorithm to perform modal analysis and estimate mode amplitude, frequency, and damping ratio on the data obtained from the Texas Synchrophasor Network. The procedure uses seven different Linear Prediction Model (LPM) orders, plus a 10 second window width that slides in steps of 1 second, to minimize the possibility of overlooking events of interest. Further, the algorithm is extended to include user defined modal characteristics thresholds, window length and step size to capture specific power system events.

TABLE OF CONTENTS

LIST OF TABLES x

LIST OF FIGURES xi

CHAPTER 1 1

INTRODUCTION 1

CHAPTER 2 3

SYNCHROPHASOR NETWORK AND LITERATURE OVERVIEW 3

2.1 Phasor Measurement Units 3

2.2 Phasor Data Concentrators 4

2.3 Synchronized Measurement Clock 5

2.4 Communication Channel 5

2.5 Texas Synchrophasor Network: 6

2.6 Literature Review 7

2.6.1 Phasor 7

2.6.2 Power system stability 8

2.6.3 Presently used FFT based screening method 8

CHAPTER 3 10

SCREENING PROCEDURE BASED ON MODAL ANALYSIS 10

3.1 Algorithm 11

3.1.1 Definition of terminologies used 11

3.1.1.1Window Length W 11

3.1.1.2 Sampling Frequency 14

3.1.1.3 Sliding Window 14

3.1.1.4 LPM order $(\boldsymbol{n})$ 15

3.1.2 Calculation of mode characteristics 15

3.1.2.1 Signal Reconstruction 17

3.1.2.2 Screening Method 17

3.2 Graphical user interface and one hour modal plots 19
3.2.1 Algorithm 20
3.2.2 Features 20
3.2.2.1 Browse 21
3.2.2.2 Load Hour 21
3.2.2.3 Load Location and Load Reference 21
3.2.2.4 File name 21
3.2.2.5 Time period to be analyzed 21
3.2.2.6 Window length 22
3.2.2.7 Step Size 22
3.2.2.8 Amplitude threshold 22
3.2.2.9 Screening Amplitude threshold 22
3.2.2.10 Frequency thresholds 22
3.2.2.11 Damping thresholds 22
3.2.2.11 Run 23
CHAPTER 4 24
RESULTS 24
4.1 Validation of Prony Algorithm: 24
4.1.1 Result I: 24th FEB 2012, 17:00:05 GMT 25
4.1.2Result II: 7th March 2012, 05:05:29GMT 25
4.1.3 Result III: 19th Jan 2012, 14:10:37 GMT 26
4.1.4 Result IV: 25th Feb 2012, 01:08:08 GMT 27
4.2 Results obtained from screening algorithm 28
4.2.1 Result I: 30th May 2011 29
4.2.1.1 Event No 1; Classical Generator Unit Trip 29
4.2.1.2 Event No 2; Transmission line Switching: 32
4.2.2 Result II: 18th Nov 2011 34
4.2.2.1 Event No 3; Opening and Reclosing of Transmission line 34
4.2.2.2 Event No 4; Unit Trip 36
4.2.2.3 Event No 5 37

4.2.3 Result III: 25th Jan 2012 39
4.2.3.1 Event No 6..... 39
4.3 Results of one hour modal plots 41
4.3.1 Result I: 25th Jan 2012, 10:00:00 GMT..... 41
4.3.1.1 Trend No 1: 10:00:00 GMT 41
4.3.1.2 Trend No 2: 07:00:00 GMT 44
CONCLUSIONS..... 45
APPENDIX..... 46
MATLAB CODE 46
REFERENCES..... 72

LIST OF TABLES

Table 1: Critical modes for MAY30, 2011, Event No 1..... 31

Table 2: Critical modes for MAY30, 2011, Event No 2..... 33

Table 3: Critical modes for Nov 18, 2011, Event No 3..... 35

Table 4: Critical modes for Nov 18, 2011, Event No 4..... 37

Table 5: Critical modes for Nov 18, 2011, Event No 5..... 38

Table 6: Critical modes for Jan 25, 2012, Event No 6..... 40

LIST OF FIGURES

Figure 1: Major Elements of a PMU..... 4

Figure 2: Hierarchy of PMUs, PDCs and super data concentrator..... 5

Figure 3: Texas Synchrophasor Network..... 7

Figure 4: Phasor representation 8

Figure 5: A 5 second window length for 25th Feb 2012, 15:00 GMT 12

Figure 6: A 10 second window length for 25th Feb 2012, 15:00 GMT 13

Figure 7: A 15 second window length for 25th Feb 2012, 15:00 GMT 13

Figure 8: A 20 second window length for 25th Feb 2012, 15:00 GMT 14

Figure 9: Snapshot of GUI 20

Figure 10: 10 second window starting from 17:00:05 GMT, 24th Feb 2012 25

Figure 11: 10 second window starting from 05:05:29 GMT, 7th March 2012 .. 26

Figure 12: 10 second window starting from 14:10:37 GMT, 19th Jan 2012 27

Figure 13: 10 second window starting from 01:08:08 GMT, 25th Feb 2012 28

Figure 14: Event No 1, beginning 03:03:05 GMT..... 29

Figure 15: Event No 1, beginning 03:03:06 GMT..... 30

Figure 16: Event No 1, beginning 03:03:07 GMT..... 30

Figure 17: Event No 1, beginning 03:03:08 GMT..... 31

Figure 18: Modal Plot at 03:03:05 GMT for a ten second window..... 32

Figure 19: Event No 2, beginning 18:51:17 GMT..... 33

Figure 20: Modal Plot at 18:51:17 GMT for a ten second window..... 34

Figure 21: Event No. 3, beginning at 05:44:26 GMT..... 35

Figure 22: Modal Plot at 05:44:26 GMT for a ten second window..... 36

Figure 23: Event No. 4, beginning at 10:57:13 GMT. 36
Figure 24: Modal Plot at 10:57:13 GMT for a ten second window 37
Figure 25: Event No. 5, beginning at 19:07:45 GMT. 38
Figure 26: Modal Plot at 19:07:45 GMT for a ten second window 39
Figure 27: Event No. 6, beginning at 07:52:25 GMT. 40
Figure 28: Modal Plot at 07:52:25 GMT for a ten second window 41
Figure 29: One hour Modal Plot for 10:00:00 GMT, 25th Jan 2012. 43
Figure 30: One hour Modal Plot for 07:00:00 GMT, 25th Jan 2012 44

CHAPTER 1

INTRODUCTION

Synchrophasors have increasingly become an important part of power system for monitoring, operation, post disturbance analysis and real time control purposes forming the wide area measurement systems (WAMS). Synchrophasor data is time stamped with the GMT clock; implying that the results obtained after investigation of synchrophasor data can be rationalized and simplified since records used to analyze events have been synchronized to the same time frame. Also, time stamping with high precision at the source makes sure that data transmission speed is no longer a critical parameter in making use of this data. All PMU measurements with the same time stamp are used to infer the state of power system at that time stamp i.e the possibility of directly estimating system state in real time rather than obtaining from system model.

The Texas Synchrophasor Network was founded at U.T. Austin in 2009 with phasor measurement unit (PMU) in McDonald Observatory in far West Texas (i.e. wind country) and a synchrophasor vector processor (SVP) at U.T. Austin [1]. With these two units it has been possible to observe voltage phase angle oscillations in the ERCOT grid and observe how they are affected by the amount of wind generation in West Texas. Since then , additional PMUs have been added at U.T. Pan American at Edinburg, at a 69 kV bus in Austin Energy, at Schweitzer offices in Boerne, and Houston, and at the Brazos Electric Power Cooperative in Waco. A number of findings have been posted on our webpage [2]. All our PMUs, except the Austin Energy 69kV unit, take their grid data from conventional 120V AC wall outlets at the rate of 30 samples/sec which are transmitted through the public network to the SVP.

Each day, 2.6 million phase angle measurements for each PMU stream into the SVP. The technical challenge is to screen these data for events of interest. Up until now, this has been done by FFT analysis, of one minute window at a time, with the objective of finding ringing events.

This thesis focuses on post event analysis using synchrophasor data from the Texas Synchrophasor Network. Post disturbance analysis is important in finding model deficiencies and implementing corresponding improvements. It helps to identify and respond to inter-area oscillations that compromise system stability. This is done by developing a screening algorithm based on modal analysis. This provides an alternate screening method in addition to the presently used Fast Fourier Transform based screening method. The work in this thesis is divided in to two parts.

The first part builds upon the key contributions of [3-6] by determining with actual data a procedure consisting of specific set of seven LPM orders, a 10 second window width, and 1-second sliding window steps, plus a set of test conditions which determine if a window has or does not have an event. This is done based on the modal parameters (amplitude, frequency and damping ratio) primarily the dominant mode characteristics and captures power system events of interest.

In the second part the algorithm is extended to capture all the modes of interest within a specific hour by developing a graphical user interface whose output is a one hour modal plot; analysis of which gives information about the system behavior.

The thesis is organized as follows. Chapter 2 describes the wide area measurement system and the Texas Synchrophasor Network followed by literature review. Chapter 3 gives explanation about the algorithm built and procedure for analyzes. Chapter 4 shows the results obtained and their interpretation, followed by conclusions.

CHAPTER 2

SYNCHROPHASOR NETWORK AND LITERATURE OVERVIEW

2.1 PHASOR MEASUREMENT UNITS

Phasor measurement units measure the magnitude and frequency of voltage, current at a very high speed (usually 30 measurements per second). Since each measurement is time stamped based on universal standard time; PMUs installed in different locations can be synchronized by aligning time stamps. The measurements obtained are transmitted either via dedicated links between specified sites, or over a switched link that is essential for the purposes of communication [7]. The PMUs are situated in power system substations and provide measurements of time-stamped positive sequence voltages and currents of all monitored buses and feeders (as well as frequency and rate of change of frequency). The measurements stored in local data storage devices can be accessed for diagnostic purposes. The phasor data is also available for real time applications in a steady stream as soon as measurements are made.

Major components of a PMU consists of a GPS receiver, Phase locked oscillator, A/D converters, Anti-aliasing filters, Modem and Phasor microprocessor as shown in Figure 1[6]

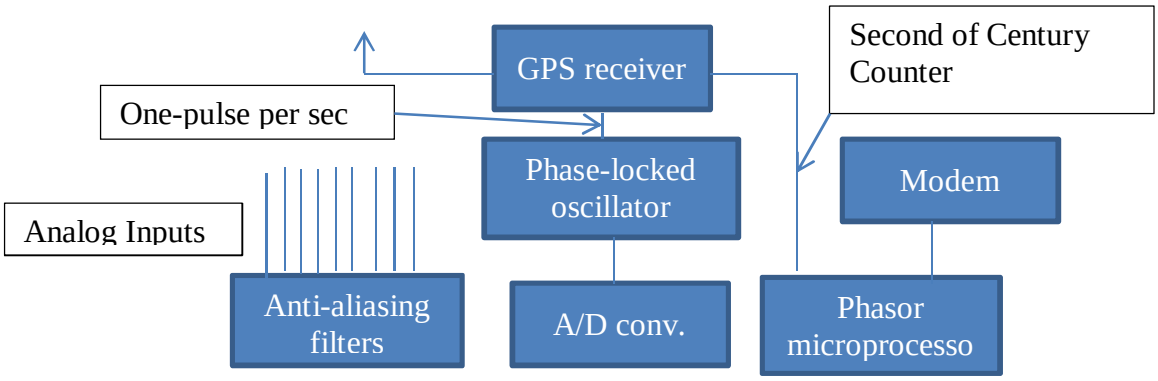

Figure 1: Major Elements of a PMU.

2.2 PHASOR DATA CONCENTRATORS

The function of Phasor Data Concentrators (PDCs) is to gather data from several PMUs, reject bad data, align the time-stamps, create a coherent record of simultaneously recorded data from a wider part of the power system and to make data stream available to application tasks with appropriate speed and latency. A Synchrophasor Vector Processor (SVP) is a PDC that includes control. It can be used to build real time distributed system integrity protection schemes. Applications include phase angle difference-based control, distributed generation control, islanding control, system control based on modal analysis, and direct state measurement. PDCs are regional in their data-gathering capability. On a system wide scale another level is contemplated known as the super data concentrator shown in Figure 2.

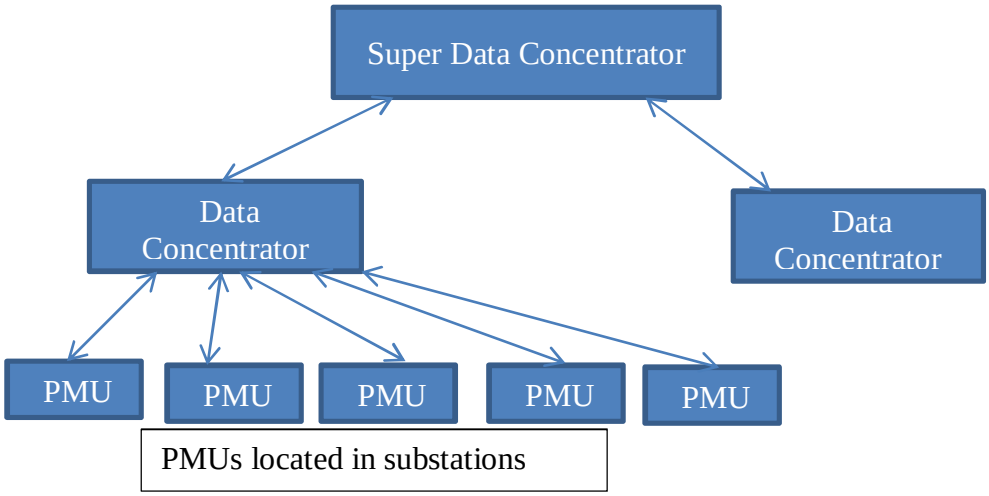

Figure 2: Hierarchy of PMUs, PDCs and super data concentrator

2.3 Synchronized Measurement Clock

The Global Positioning System (GPS) provides time tags to the PMU measurements. The GPS satellites keep accurate clocks which provide one-pulse per second. The time they keep is known as GPS time which does not take into account the Earth's rotation. Corrections to the GPS time are made in the GPS receivers to amount for this difference (leap-second correction) so that the receivers provide UTC clock time. The identity of the pulse is defined by the number of seconds since the time that the clocks began to count (January 6, 1980). However, the PMU standard uses UNIX time base with a "second-of -century" (SOC) counter which began counting at midnight on January1, 1970.

2.4 Communication Channel

The communication link is bidirectional. Most of the data flow is upward in hierarchy, although there are some tasks which require communication capability in the reverse direction[8]. Two aspects of data transfer namely channel rate and latency affect the communication task. Channel capacity is the measure of data rate (in kilobits/second

or megabits/second). Channel capacity is rarely a limiting factor in most applications as channel volume created by PMUs is quite modest. Latency is the time lag between the time at which the data is created and when it is available for desired application. Applications for real-time control require very small latency, as compared to post-event analysis applications which are not affected by large delays in transferring data. Communication links used by PMUs based WAMs can be both wired and wireless.

2.5 TEXAS SYNCHROPHASOR NETWORK:

Texas synchrophasor network consists of PMUs installed in UT Austin, McDonald Observatory in Fort Davis, UT Pan Am, Houston and Boerne all within Electric Reliability Council of Texas (ERCOT) [1,2]. Each PMU is powered by 120V wall outlet and data is recorded with a frequency of 30 samples per second. These PMUs send data to the Austin Synchrophasor vector processor over the public internet.

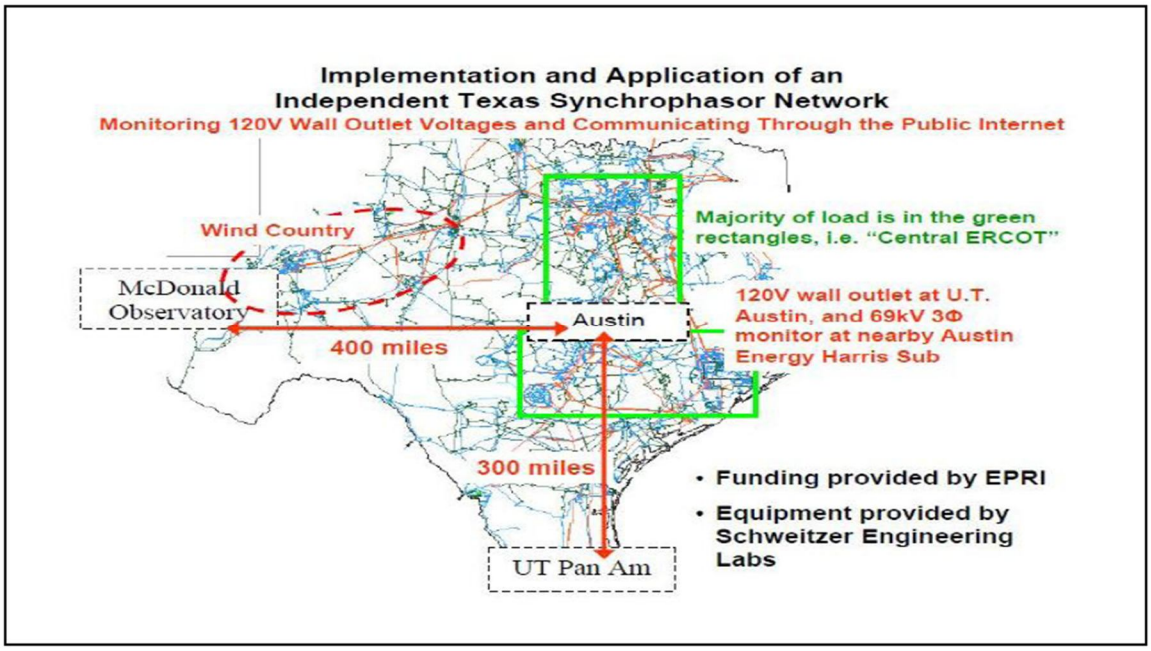

Figure 3: Texas Synchrophasor Network

2.6 LITERATURE REVIEW

2.6.1 Phasor

A sinusoidal voltage or current at constant frequency is characterized by two parameters, a maximum value and a phase angle [9]

$$x(t) = X_m \cos(\omega t + \varphi) \qquad (2.1)$$

Where, X_m is the amplitude of the sine wave,

ω is the frequency in radians/sec

φ is the phase angle measured with respect to a reference

Measurement obtained from PMUs are time stamped implying that the phase angle of all measurements have the same reference.

The sinusoid of above equation is represented by a complex number X known as its phasor representation:

$$X = \frac{X_m}{\sqrt{2}} \angle \varphi \qquad (2.2)$$

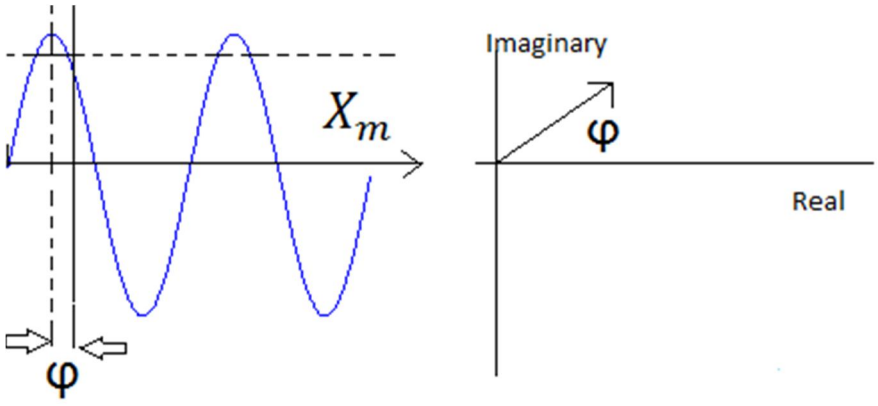

Figure 4: Phasor representation

2.6.2 Power system stability

Transient stability is the ability of power system to maintain synchronism when subjected to severe transient disturbance, whereas small signal stability refers to the system capability to maintain synchronism after small disturbance. The power transferred between two buses in power system depends on the rotor angle δ which is the difference of bus voltage angle of the two buses. Rotor angle forms an important in analyzing the electromechanical oscillations inherent in the power system. PMU measurements help make conclusions about power system stability through post-disturbance analysis.

2.6.3 Presently used FFT based screening method

Presently the visual basic program PMU_Waveform_Analyzer developed by professor W. Mack Grady is being used to screen events of interest[2]. This program uses the '.csv' files containing the data obtained from the PMUs. It consists of voltage and angle measurements of the locations within the Texas Synchrophasor network. The

program allows to select the day and hour to be analyzed and “screen day for ringdowns” features generates a data file containing information about power system disturbances captured during that hour.

CHAPTER 3

SCREENING PROCEDURE BASED ON MODAL ANALYSIS

Modal analysis provides considerable insight into stability properties of power system. Time-synchronized measurements provide rich information for estimating power system's electromechanical modal properties which is critical information for improved operational reliability of interconnected grids. A given mode's properties are described by its frequency, damping, and amplitude. Modal frequencies and damping are useful indicators of power system stress, usually declining with increased load or reduced grid capacity. Modal amplitude provides critical information for operational control actions [10]. Mode shape may be used to determine generator and/or load-tripping schemes to improve the damping of a dangerously damped mode. The optimization involves minimizing load shedding and maximizing improved damping. The two enabling technologies for such real-time applications are a reliable real time-synchronized measurement system and accurate modal analysis signal processing algorithms. In terms of application, modal frequency and damping estimation algorithms fall into the two categories of ringdown analyzers and mode meters. A ringdown analysis tool operates specifically on the ringdown portion of the response i.e the first several cycles of the oscillation (5-20s). A mode meter is an automated tool applied to any portion of response: ambient, transient, or both that estimates modal properties continuously and without reference to any exogenous system input.

This thesis uses prony analysis as a signal processing tool to extract modal parameters from the PMU measurements. These modal parameters form the basis of screening procedure.

3.1 ALGORITHM

Prony analysis is a method of fitting a linear combination of exponential terms to a signal $x(t)$ with the form shown in (3.1).

$$x(t) = \sum_{i=1}^{n} A_i e^{\alpha_i t} \cos(2\pi f_i t + \theta_i) \qquad (3.1)$$

The purpose of prony algorithm is to estimate the eigenvalues associated with the system states. These eigenvalues contain information about frequency and damping ratio. To accomplish this it utilizes a linear prediction model that forecasts future values of a signal from a linear combination of past values [11]. Further amplitude is then estimated from the Linear Prediction Model (LPM) built. Each term in the above equation has four elements: magnitude A_i, damping constant α_i, frequency f_i, and phase angle θ_i [6]. Each exponential component is viewed as a unique mode of the original signal $x(t)$. The total number of modes n calculated is the order of the LPM. However, since estimation of data is an ill conditioned problem, one algorithm could perform completely different on different signals. Factors which affect the accurate estimation by Prony algorithm are window length, sampling time and linear prediction model order number. Tuning of these prony parameters are crucial for efficient modal analysis. This work proposes variable window length, equal sampling time and a variable order LPM model for accurate estimation of dominant modes.

3.1.1 Definition of terminologies used

3.1.1.1Window Length W

The time interval over which this algorithm is performed is called the window length W. This can be interpreted as the total number of data samples taken corresponding to that contained inside W seconds. Choosing an appropriate window length is crucial for the algorithm to reconstruct the new signal close to the original

contained in W and also to accurately determine the modes of interest. In transient stability studies the study period of interest is usually limited to 3-5 seconds following the disturbance, although it may extend to about 10 seconds for large systems with dominant interarea modes of oscillation. Most of the work based on prony algorithm uses a 10 second analysis window for the same reason [12]. In this work a window length of 10 seconds is used. This was concluded by the above reason and also tested with algorithm application for various window length. It was found as shown by the results below that a window length of less than or equal to 10 second gives satisfactory results. A 10 second window was thus chosen to ensure the power system events are captured completely. Below are the results from the day of 25th of Feb 2012, 15th hour GMT, the window length is increased from 5 to 25 seconds starting at 15:00 GMT.

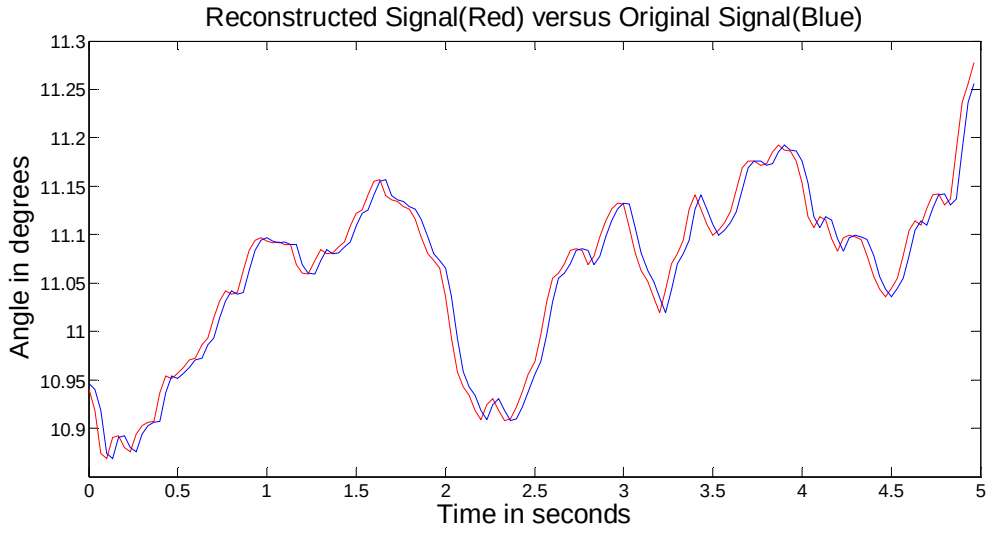

Figure 5: A 5 second window length for 25th Feb 2012, 15:00 GMT

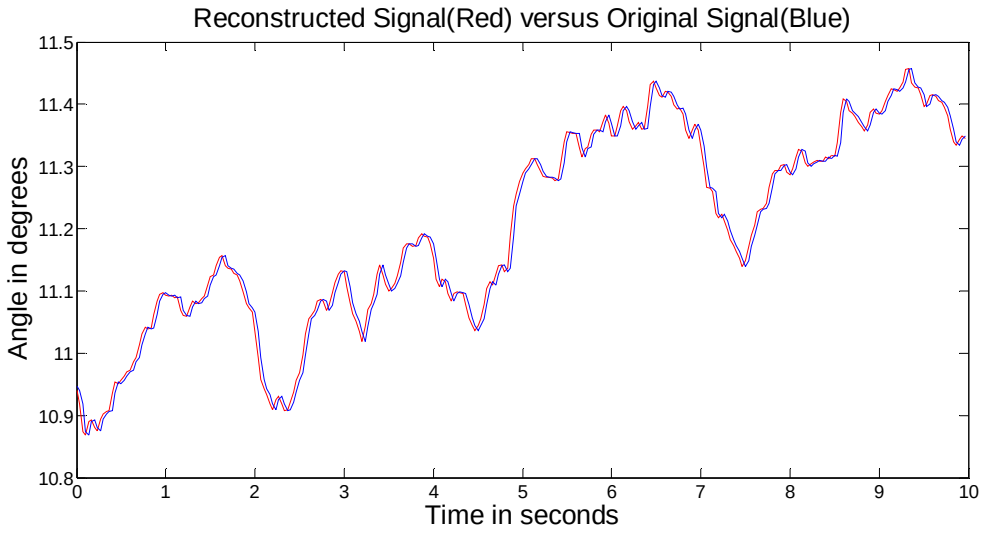

Figure 6: A 10 second window length for 25th Feb 2012, 15:00 GMT

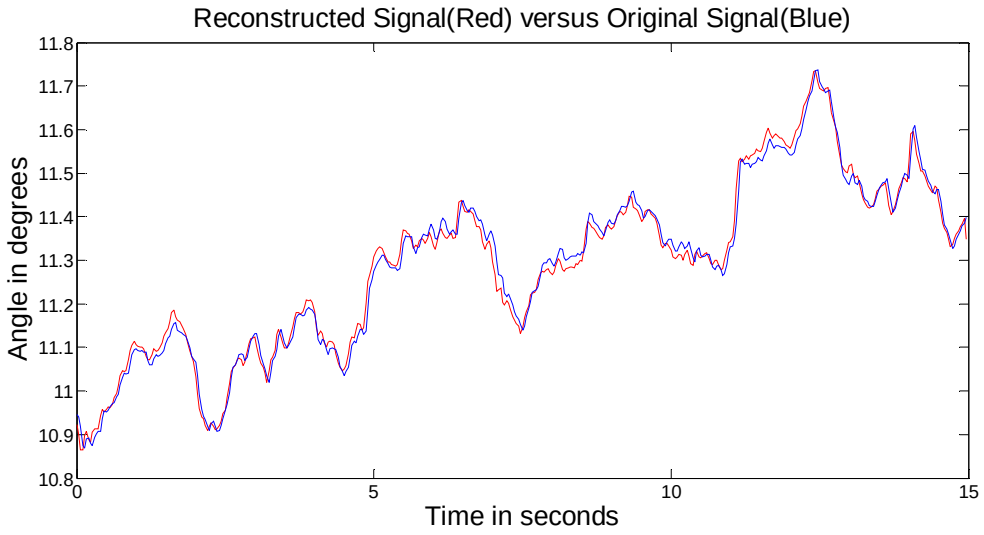

Figure 7: A 15 second window length for 25th Feb 2012, 15:00 GMT

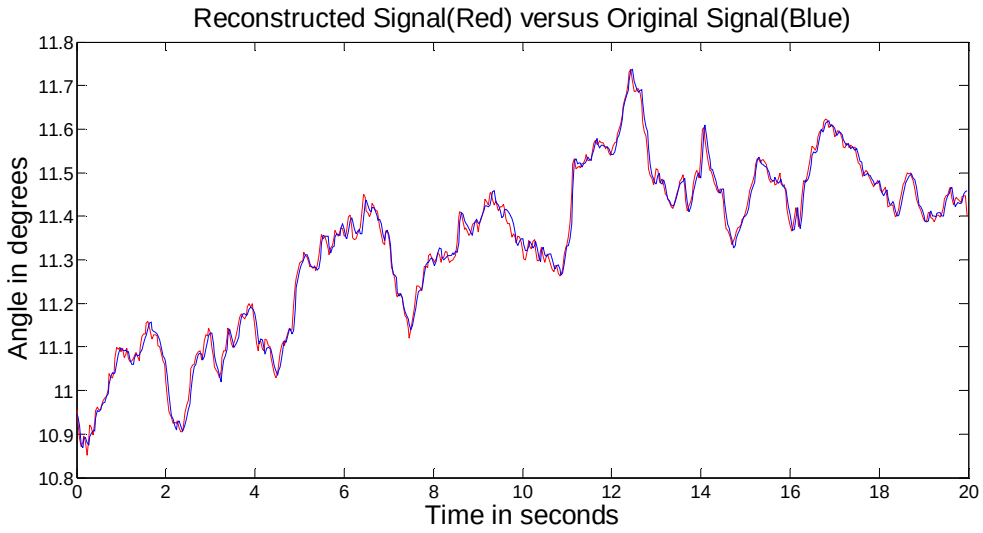

Figure 8: A 20 second window length for 25th Feb 2012, 15:00 GMT

From the above results it was concluded to take a 10 second window for future analysis purposes.

3.1.1.2 Sampling Frequency

The data is received at a sampling frequency (f) of 30 samples/sec. The total number of data samples N is the product of window length W and the sampling frequency. We denote $T = 1/f$ as the time period between two samples.

3.1.1.3 Sliding Window

The concept of sliding and overlapping window is used with an overlapping step size of 1 second for purposes of screening power system events. This ensures that the start and stop timings of power system events are captured accurately. Later when the program is extended to a build a graphical user interface, both the window length and step size can be varied.

3.1.1.4 LPM order (n)

As described above LPM order is the total number of modes obtained within a window length W. Recommendations for LPM order (n) are that it should be one-third to one-half of the total number of data samples (N) [3-6]. Choice of (n) is discussed later in the thesis.

3.1.2 Calculation of mode characteristics

A mode is characterized by its amplitude, frequency, damping ratio and phase. The following algorithm first describes the calculation of these parameters followed by the screening procedure.

Let $x[1], x[2] \ldots \ldots . x[N]$ denote the total number of data samples in one window length . Following procedure in [3], we first calculate the LPM coefficients a_i.

Estimation of LPM coefficients a_n is crucial for the derivation of frequency, damping, amplitude and phase angle. To estimate these coefficients accurately, many algorithms can be used. A matrix representation of signal at various sample times can be formed by sequentially writing the linear prediction of x_{N+1} repetitively given by (3.2)

$$x_{N+1} = a_1 x_N + a_2 x_{N-1} + \quad \ldots\ldots \ \ldots\ldots \quad + a_n x_{N-n+1} \tag{3.2}$$

By inverting the matrix representation, the linear coefficients a_n can be obtained given by (3.3)

$$\begin{bmatrix} x[n] & . & . & x[1] \\ . & . & . & . \\ . & . & . & . \\ x[N-1] & . & . & x[N-n] \end{bmatrix} \begin{bmatrix} a_1 \\ . \\ . \\ a_n \end{bmatrix} = \begin{bmatrix} x[n+1] \\ . \\ . \\ x[N] \end{bmatrix} \tag{3.3}$$

$$Xa = x \tag{3.4}$$

$$a = X^{-1}x \tag{3.5}$$

Since X is a rectangular matrix, in Matlab it is inverted using the 'pinv' function, which calculates the Moore-Penrose pseudo inverse of X. The calculation is based on singular value decomposition (SVD) of the matrix X and any singular values less than 'tol' is taken as zero.

Given the a_i's, we calculate the roots of characteristic polynomial (3.6), which has coefficients $1, a_1, a_2 \ldots\ldots\ldots a_n$, to yield z_i.

$$z^n - a_1 z^{n-1} - a_2 z^{n-2} \ldots\ldots\ldots - a_n z^0 = 0 \qquad (3.6)$$

Calculation of z_i is necessary to obtain the eigenvalues which are related to z_i by the relation:

$$e^{\lambda_i} = z_i \qquad (3.7)$$

For any i^{th} eigenvalue, the real part contains information about the damping constant, and the imaginary part gives the value of natural frequency, i.e $\lambda_i = \alpha_i + j\omega_i$, where α_i is the damping constant and ω_i is the natural frequency in rad/s.

Damping constant α_i, damping ratio d_i, and frequency f_i (in Hz) is then calculated as,

$$\alpha_i = \frac{Real(logz_i)}{T}, \quad d_i = \frac{-\alpha_i}{\sqrt{{\alpha_i}^2 + {\omega_i}^2}} \qquad (3.8)$$

$$f_i = \frac{Im(logz_i)}{2\pi T} \qquad (3.9)$$

To calculate modal amplitudes A_i, we first calculate B_i 's as the solution matrix of (3.10) solved in a least squared sense.

$$\begin{bmatrix} {z_1}^0 & . & . & {z_n}^0 \\ . & . & . & . \\ . & . & . & . \\ {z_1}^{N-1} & . & . & {z_n}^{N-1} \end{bmatrix} \begin{bmatrix} B_1 \\ . \\ . \\ B_n \end{bmatrix} = \begin{bmatrix} x[1] \\ . \\ . \\ x[N] \end{bmatrix} \quad (3.10)$$

A_i's are then obtained by (3.11)

$$A_i = 2 * |B_i| \quad (3.11)$$

Out of the total number of modes obtained (n), we discard the ones that have frequency less than or equal to zero. The dominant mode is the one with the greatest A_i. The frequency and damping ratio associated with this mode are the characteristic parameters of the signal contained in window .

3.1.2.1 Signal Reconstruction

Once the B_i's are computed, we then calculate the estimated data points $\hat{x}[1], \hat{x}[2] \dots \dots \dots \dots \hat{x}[N]$ using (3.12)

$$\hat{x}[k] = \sum_{i=1}^{n} {B_i}^k \, {z_i}^k \ , \qquad k = 1 \ to \ N \quad (3.12)$$

To compare the new reconstructed signal with that of original signal we use Signal to noise ratio (SNR) which is a goodness measure of the accuracy of the reconstructed signal [3], and it is given by (3.13)

$$SNR = 10 log_{10} \left(\frac{\sum_{i=1}^{N} x[i]^2}{\sum_{i=1}^{N} (x[i] - \hat{x}[i])^2} \right) dB \quad (3.13)$$

As pointed out by [7], an SNR greater than 40dB indicates that the estimates $\hat{x}[1], \hat{x}[2] \dots \dots \dots \dots \hat{x}[N]$ are sufficiently accurate.

3.1.2.2 Screening Method

As stated above there is no defined rule for selection of LPM order (n), except that it should be between one-third to one-half of total number of data samples [3-6].

Selection of this parameter is crucial for effective modal analysis [6]. Signal to Noise Ratio or ability of algorithm to reconstruct the original signal changes with the Linear Prediction Model order. Thus the algorithm proposed here iteratively calculates modes changing LPM order such that SNR and amplitude of the dominant modes are within acceptable range. The procedure was tested for a variety of different events, and based upon the testing the following seven values of n are recommended

$$\left\{\frac{N}{2}-20,\frac{N}{2}-15,\frac{N}{2}-10,\frac{N}{2}-4,\frac{N}{2}+6,\frac{N}{2}+10,\frac{N}{2}+17\right\} \quad (3.14)$$

The dominant mode for each of the seven values of LPM order (n) is identified for each 10 second window W. The window W is considered to have an event of interest if it meets all of the following criteria:

- Amplitude of dominant mode is greater than 1º.
- Frequency of dominant mode is less than 2 Hz.
- Dominant mode damping ratio is less than 0.2.
- SNR is greater than 40dB.

When a window W meets all the above four criteria, for all seven values of n, the dominant mode for the window W that is reported is the one corresponding to the LPM order (n) that gave the maximum SNR. A justification of the above process follows.

Amplitude greater than 1º; this criterion was selected by testing the data first for a cut-off threshold of 0.2º which was raised to 0.5º and then to 1º. It was found that a 1º threshold identified the major events and so was selected as the final cut-off threshold.

Frequency less than 2 Hz; this criterion was selected since the dominant mode in ERCOT is in the 0.5-1.0 Hz range.

Damping ratio less than 0.2; power system events of interest to us are lightly damped, so first a threshold of 0.3 was selected. After analysis and testing on a number of days it was found that this threshold does not provide good screening results, so the threshold was tightened to 0.2, which proved successful.

The above algorithm coded in Matlab is run for a whole day, and a '.xls' file is generated which contains information about the filtered modes. This file reports the hour, start time and stop time of the power system event (in seconds) and the characteristics of the dominant mode i.e its amplitude, frequency and damping ratio. Looking back at the window W reported by the '.xls' file it was observed that each of them corresponded to a power system event. This is discussed in more detail in results.

3.2 GRAPHICAL USER INTERFACE AND ONE HOUR MODAL PLOTS

The purpose of building a GUI was to provide a platform for analyzing hours of interest with user defined thresholds. It is to be noted that in the previous case the screening algorithm was developed based on analyzing a number of days and the modal characteristics (amplitude, frequency and damping ratio) were fixed based on the observations. However, this platform provides flexibility in terms of analyzing and looking at desired modal characteristics (since they are user entered). Also, it should be noted that the screening algorithm is based on the information solely restricted to the dominant mode. However, here we look at all the modes that meet the user defined thresholds in the hour which is being analyzed. This is useful in drawing conclusions particularly in identifying certain trends which is later explained with the help of results in Chapter 4. The program creates a '.xls' file at the end which contains information about the dominant mode filtered according to the screening criterion. Also, the

parameters for the screening algorithm are user-defined. Below is a snapshot of the GUI build.

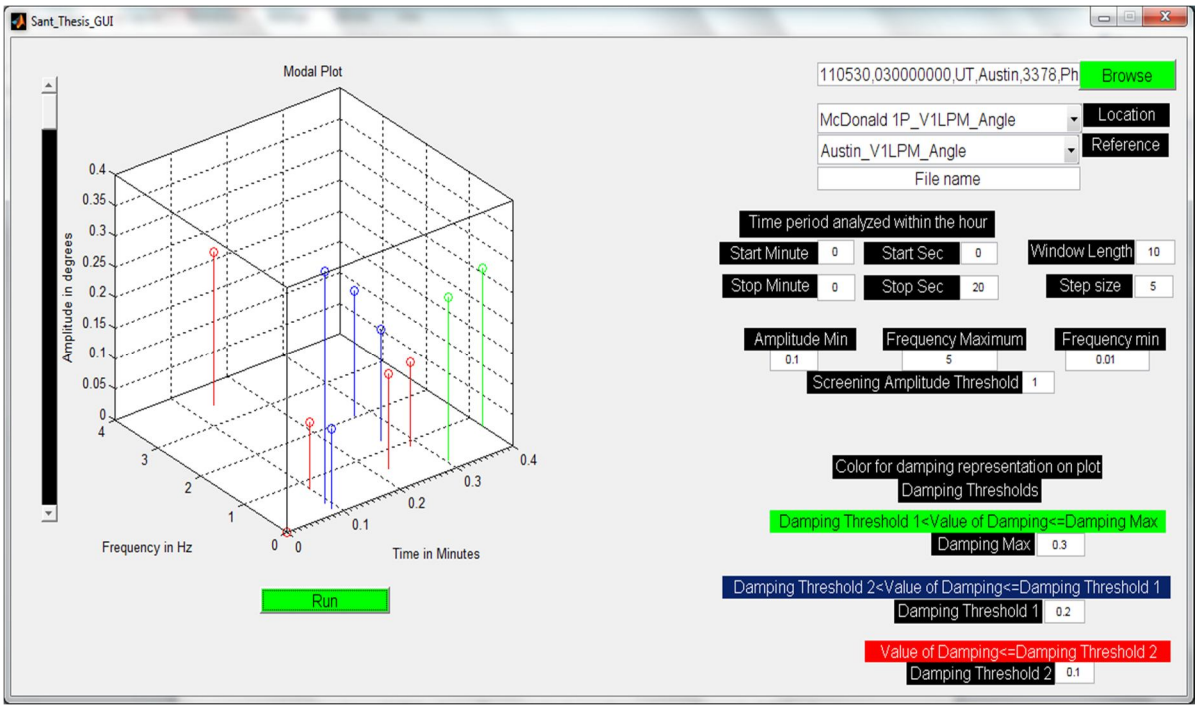

Figure 9: Snapshot of GUI

3.2.1 Algorithm

The screening algorithm is the same as above. However, for displaying all the modes of the hour under analysis; we first obtain all the modes corresponding to the seven values of LPM order n. Then we take the modes corresponding to the LPM order n which gives the maximum SNR. These modes are then filtered according to the thresholds provided by user and plotted on a 3D graph in the form of stem points.

3.2.2 Features

The features of this interface are described below:

3.2.2.1 Browse

This is a ‘push button’ and on pressing it opens the Matlab folder from which the ‘.csv’ file corresponding to the hour to be analyzed can be selected. The ‘.csv’ file contains the data from the synchrophasor locations with each column corresponding to one synchrophasor location.

3.2.2.2 Load Hour

This is an ‘edit box’ that displays the ‘.csv’ filename selected by the Browse button.

3.2.2.3 Load Location and Load Reference

These are two ‘pop-up-menus’. When the hour has been selected all the synchrophasor locations are loaded in these two menus. The location to be analyzed is selected using the ‘Load Location’ menu and the reference location is selected using the ‘Load Reference’ menu.

3.2.2.4 File name

This is an ‘edit button’ that takes in input a file name by which the user wants to store the ‘.xls’ file generated by the algorithm that contains information about the dominant modes that met the screening criterion.

3.2.2.5 Time period to be analyzed

The algorithm can be run for a whole hour, alternatively a particular time period within the hour can also be analyzed. This is achieved through four ‘edit buttons’ namely; start minute, start second, stop minute and stop second. These can be used to fix the time period of analysis. For a whole hour the value of each of these is:

- Start Minute; 0
- Start Second; 0

- Stop Minute; 59
- Stop Second; 59

3.2.2.6 Window length

This is an edit button that adjusts the window length W.

3.2.2.7 Step Size

The algorithm takes step-size through this edit button. As described previously, step size is the number of seconds over which the sliding window moves forward to analyze the next window length W.

3.2.2.8 Amplitude threshold

This defines the minimum amplitude with which each of the modes is compared. Only the modes with amplitude greater than this value are retained and plotted.

3.2.2.9 Screening Amplitude threshold

This defines the amplitude threshold for only the dominant mode used for the screening algorithm that was described in the previous section.

3.2.2.10 Frequency thresholds

The range of frequency is fixed using the Frequency Maximum and the Frequency Minimum edit buttons. The frequency minimum edit button is usually set close to zero since negative frequencies have no meaning.

3.2.2.11 Damping thresholds

There are three damping thresholds provide for the purpose of distinguishing highly damped modes from modes with low damping. Modes having a damping ratio between ‘Damping maximum’ and ‘Damping threshold 1’ are shown by green color. Modes having damping ratio between ‘Damping threshold 1’ and ‘Damping threshold 2’

are shown by blue color and modes having damping ratio below than ‘Damping threshold 2’ are shown as red.

3.2.2.11 Run

This is a ‘push button’ and after pressing ‘run’ button the graph shows all the filtered modes. Looking at the mode features conclusions can be drawn about the particular time period.

CHAPTER 4

RESULTS

The results are divided into three parts, the first part test the ability of algorithm to reconstruct the original signal accurately. This is judged based on the signal to noise ratio. The second part shows the results obtained from the screening procedure developed. The third part shows the application of GUI for post-disturbance analysis.

4.1 VALIDATION OF PRONY ALGORITHM:

As stated previously, this algorithm is a linear prediction model based method that extracts the modal information of the signal under consideration. The modes thus obtained can be used to reconstruct the original signal and the extent to which the newly reconstructed signal matches the original signal helps in validating the accuracy of the algorithm. The new signal is constructed using Eq. 3.12

$$\hat{x}[k] = \sum_{i=1}^{n} {B_i}^k \, {z_i}^k \ , \qquad k = 1 \; to \; N$$

Where,

N is the total number of data samples

n is the LPM order number

k denotes the newly reconstructed k^{th} data point out of the total N data points

As pointed out by [11] prony analysis cannot obtain accurate estimation when SNR is less than 40db. We see from the results below that for each of the window W under consideration the SNR meets the aforementioned criterion.

4.1.1 Result I: 24th FEB 2012, 17:00:05 GMT

We analyze phase angle difference between McDonald Observatory in West Texas and University of Texas at Austin. The SNR for the window length W of 10 seconds starting from 17:00:05 GMT was obtained as 83.063dB which is well above 40dB and validates the prony algorithm

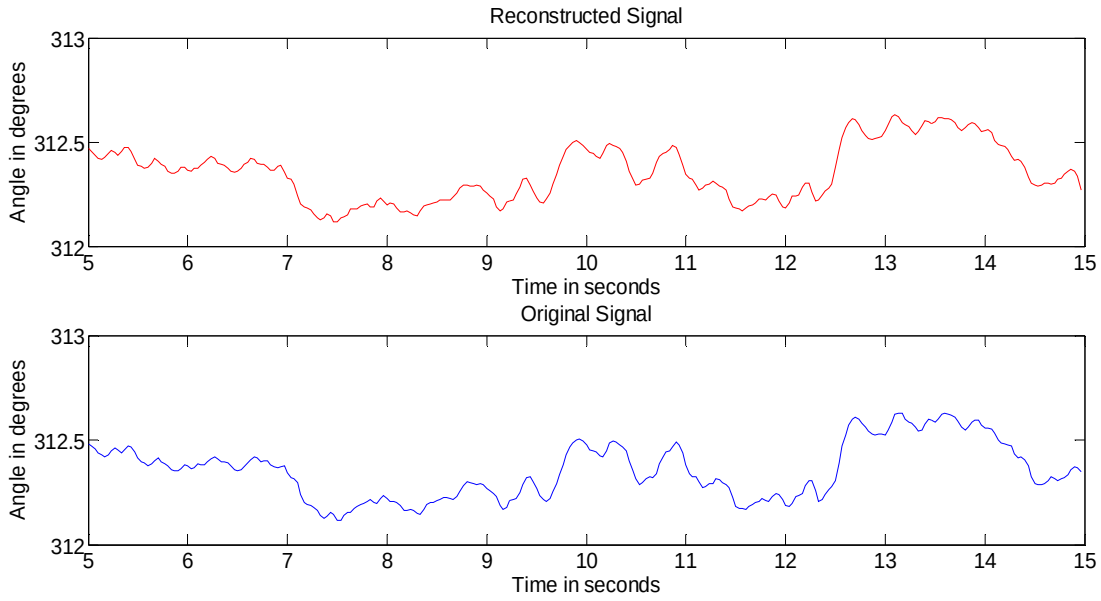

Figure 10: 10 second window starting from 17:00:05 GMT, 24th Feb 2012

4.1.2Result II: 7th March 2012, 05:05:29GMT

The SNR for the window length W of 10 seconds starting from 05:05:29 GMT was obtained as 60.99dB which is above 40dB. The angle difference was between McDonald Observatory and UT Austin.

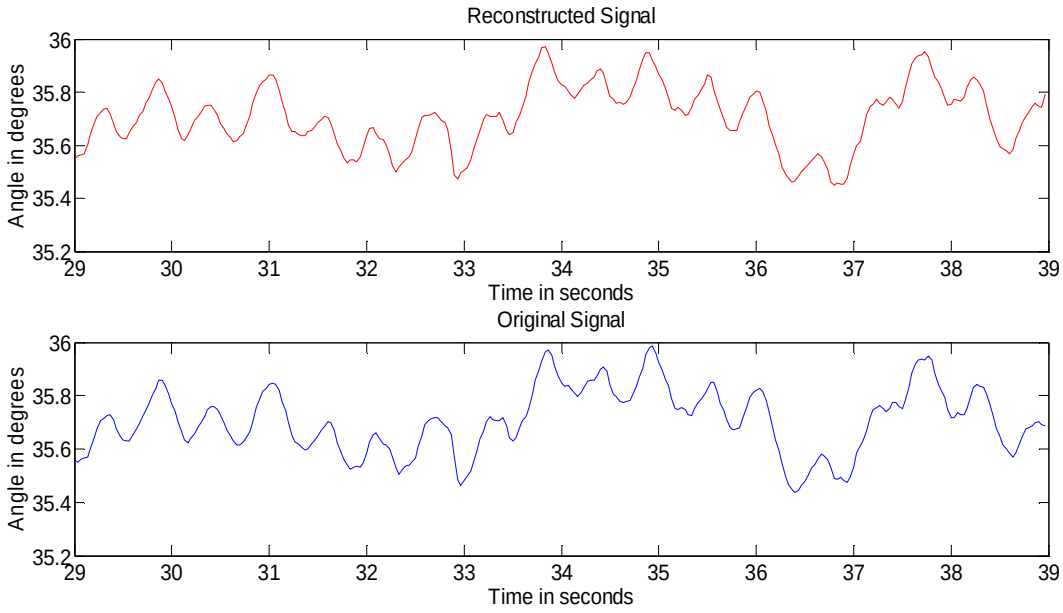

Figure 11: 10 second window starting from 05:05:29 GMT, 7th March 2012

4.1.3 Result III: 19th Jan 2012, 14:10:37 GMT

The angle difference between UT Pan Am and UT Austin is analyzed for the 10 second window starting at 14:10:37 GMT. The SNR was obtained as 56.105dB.

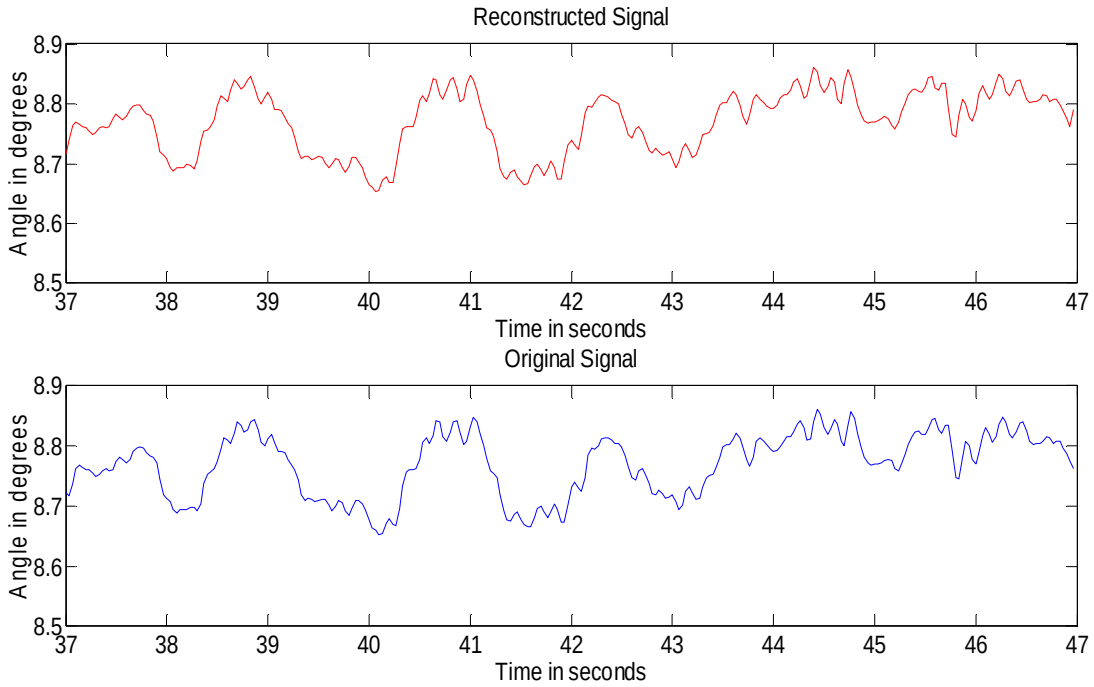

Figure 12: 10 second window starting from 14:10:37 GMT, 19th Jan 2012

4.1.4 Result IV: 25th Feb 2012, 01:08:08 GMT

Below is the 10 second window starting at 01:08:08 GMT for the phase angle difference between UT Pan Am and UT Austin with an SNR of 63.53 dB.

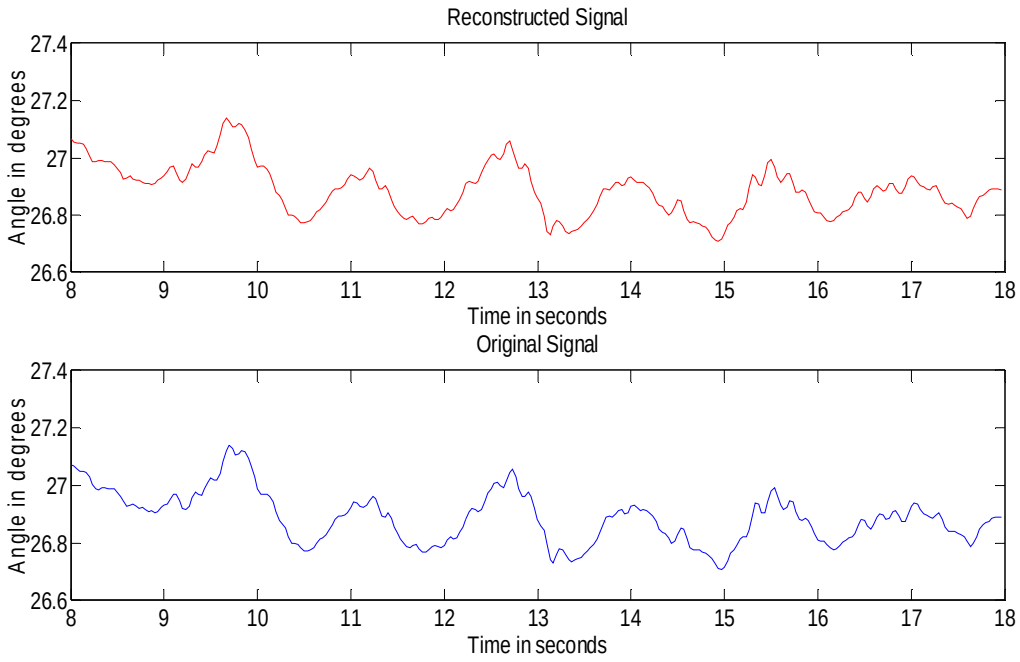

Figure 13: 10 second window starting from 01:08:08 GMT, 25th Feb 2012

From the above results it is clear that the algorithm can reconstruct the original signal successfully implying the accuracy of the modes characteristics (amplitude, frequency and damping ratio) and thus these modes can be used with reasonable confidence for screening purposes.

4.2 RESULTS OBTAINED FROM SCREENING ALGORITHM

A number of days were tested using the screening procedure and following are some of the interesting power system events obtained. All the waveforms for the results below show actual waveform as a solid line and prony reconstructed waveform as a dotted line. However, the two are almost indistinguishable.

4.2.1 Result I: 30th May 2011

The voltage angle difference between U.T. Pan Am and U.T. Austin was analyzed and the algorithm runs for a time period of all 24 hours of this day. The screening procedure identified two major events.

4.2.1.1 Event No 1; Classical Generator Unit Trip

The screening procedure identified event no 1 in four consecutive 10 second sliding windows (shown in Figure 14-Figure 17).

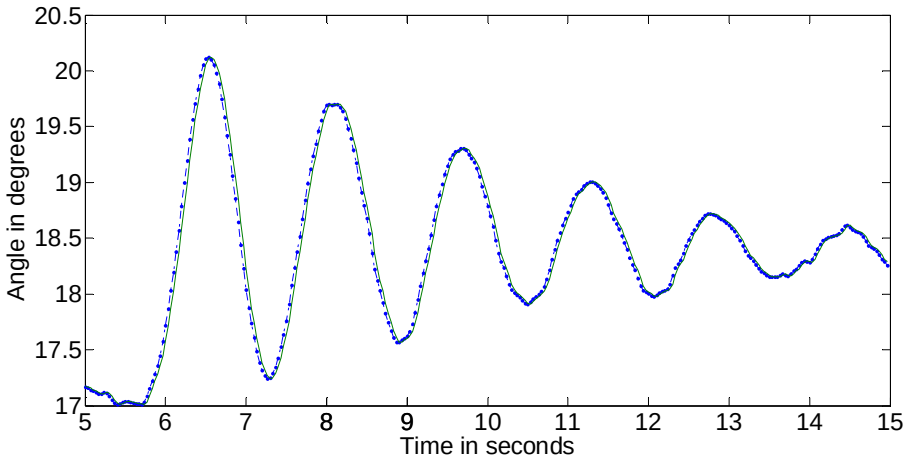

Figure 14: Event No 1, beginning 03:03:05 GMT.

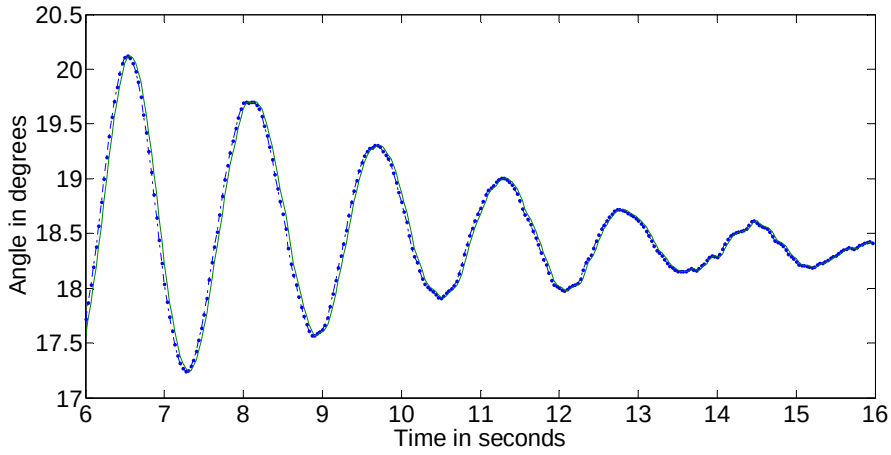

Figure 15: Event No 1, beginning 03:03:06 GMT

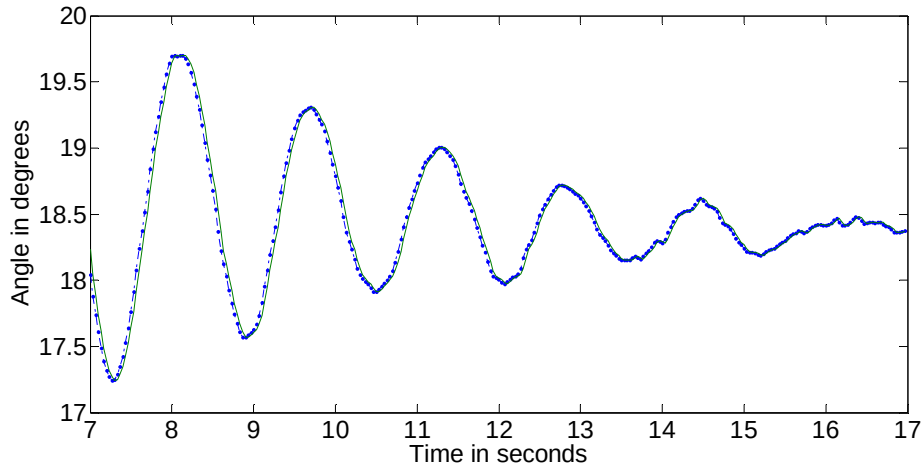

Figure 16: Event No 1, beginning 03:03:07 GMT

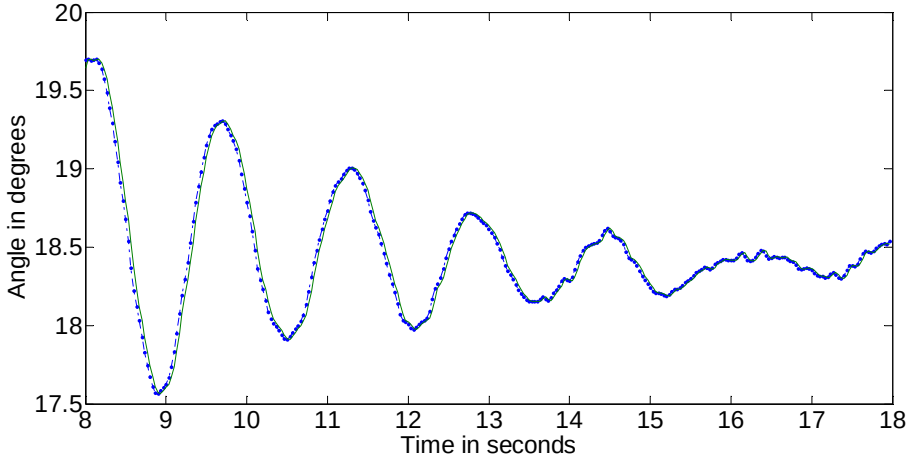

Figure 17: Event No 1, beginning 03:03:08 GMT

The characteristics of the dominant mode were reported as a '.xls' file. Below is the table (Table 1) extracted from the '.xls' file for each consecutive ten second window.

FIG NO	HOUR:MIN	START SEC	AMP (DEG)	FREQ (HZ)	DAMPING RATIO	SNR (DB)
14	3:03	5	2.43	0.636	0.063	62.5
15	3:03	6	2.08	0.624	0.072	62.6
16	3:03	7	1.78	0.633	0.072	64.7
17	3:03	8	1.19	0.632	0.063	66.7

Table 1: Critical modes for MAY30, 2011, Event No 1.

Figure 14 contains the entire event, and the dominant mode has amplitude of 2.43º, frequency of 0.63 Hz, and a damping ratio of 0.063. Looking at the corresponding modal information given by Figure 18 below it can be seen that two other modes are significant. The fact that the three significant modes have approximately the same

characteristics is strong evidence that this event is a classic ringdown especially in offline processing, where graphical inspection is not possible.

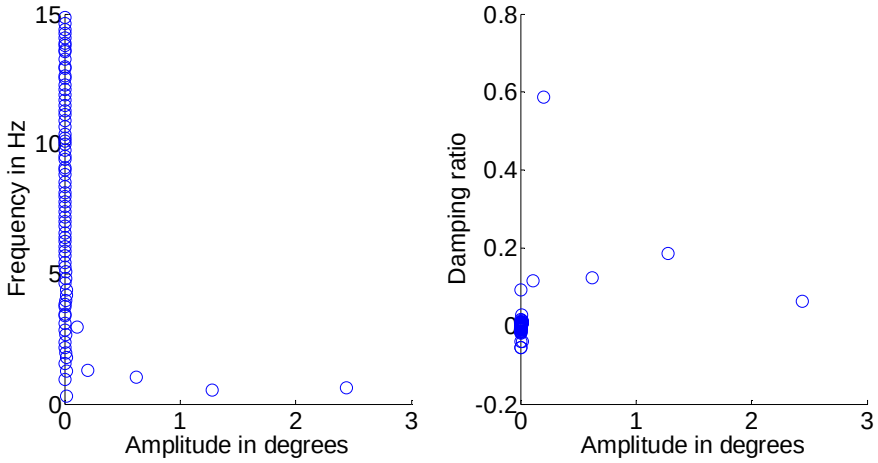

Figure 18: Modal Plot at 03:03:05 GMT for a ten second window.

The fact that the pre and post-event phase angle rises from 17.1º to 18.5º (Figure 14) indicates a unit trip remote to West Texas.

4.2.1.2 Event No 2; Transmission line Switching:

This event was identified in one sliding window starting at 18:51:17 GMT.

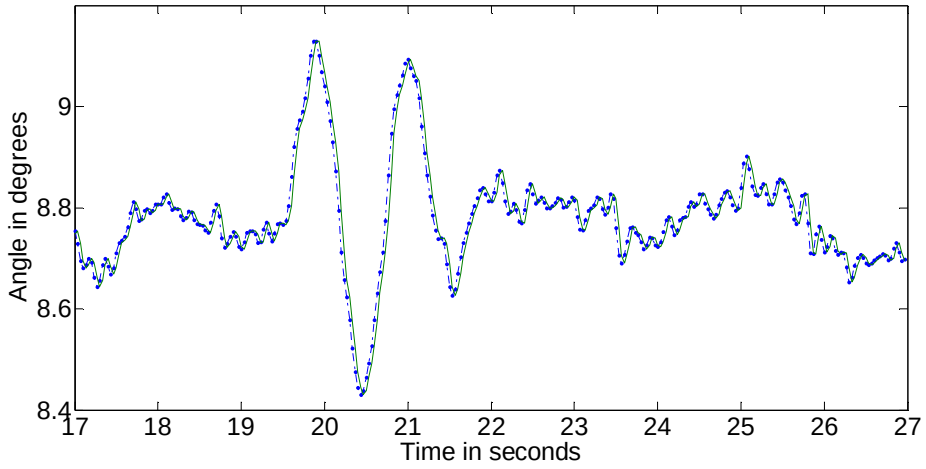

Figure 19: Event No 2, beginning 18:51:17 GMT.

The critical modes for this event are given by Table 2 below. Table 2 and the modal plot given by Figure 20 show that the frequency corresponding to the dominant mode is 0.75Hz. From Figure 19 we observe that the phase angle before and after the disturbance is the same and the ringdown is not a damped sinusoid, but rather it is a sudden jump of angle to a new value and back to its original value in a duration of less than 2 seconds (from 20th to 22nd second). This implies that the event is most likely caused by transmission line opening and closing two seconds later.

FIG NO	HOUR:MIN	START SEC	AMP (DEG)	FREQ (HZ)	DAMPING RATIO	SNR (DB)
19	18:51	17	1.07	0.748	0.148	72.0

Table 2: Critical modes for MAY30, 2011, Event No 2.

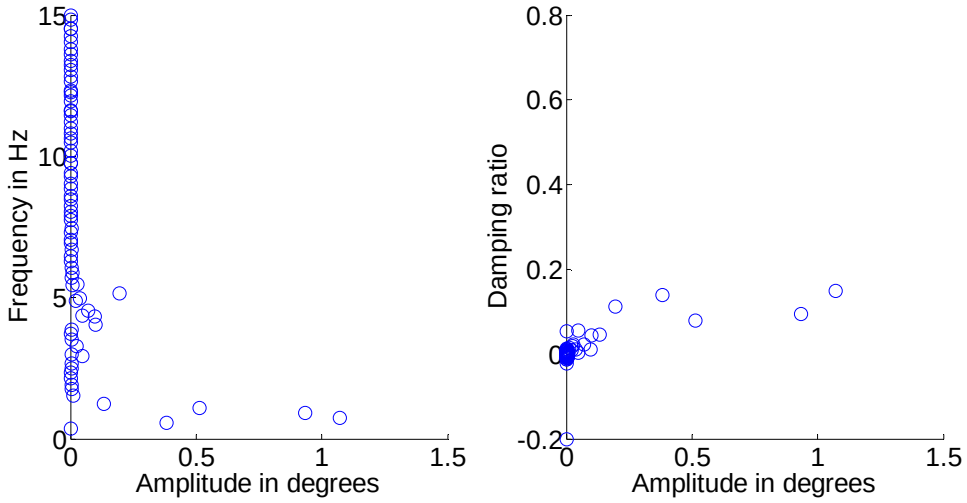

Figure 20: Modal Plot at 18:51:17 GMT for a ten second window.

4.2.2 Result II: 18th Nov 2011

It was observed that 18th Nov 2011 had high wind penetration in West Texas. So this was chosen as a good test day to verify the results. The angle difference between McDonald Observatory with respect to U.T. is analyzed.

4.2.2.1 Event No 3; Opening and Reclosing of Transmission line

Figure 21 shows the window length identified for this event (05:44:26 GMT to 05:44:36 GMT).

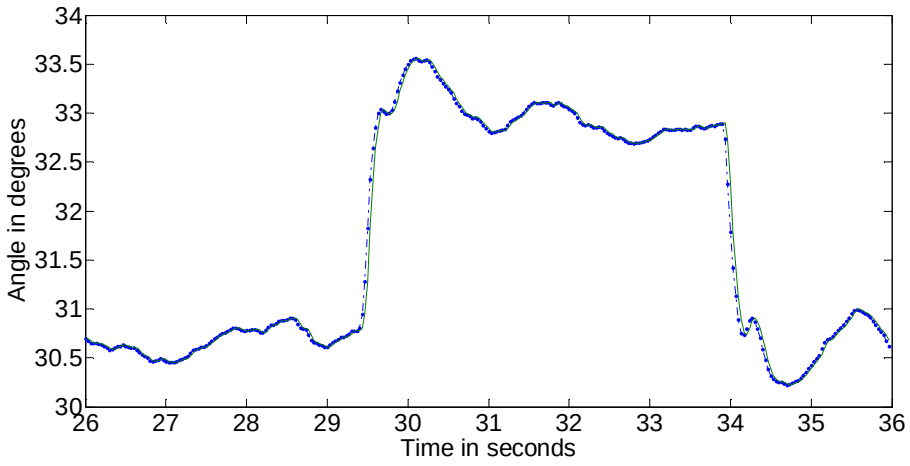

Figure 21: Event No. 3, beginning at 05:44:26 GMT.

Table 3 and the modal plot (Figure 22) show the frequency and damping ratio corresponding to dominant mode to be 0.108 Hz and 0.066 respectively. This frequency is far below what we normally see in ERCOT. During this event we see that the angle jumped to a new value of around 33.5º, followed by a damped sinusoid and the angle went back to its initial value of around 31º. Thus, this is most likely opening and reclosing of a transmission line as the initial and final value of phase angle are the same.

FIG NO	HOUR:MIN	START SEC	AMP (DEG)	FREQ (HZ)	DAMPING RATIO	SNR (DB)
21	05:44	26	1.92	0.108	0.066	51.1

Table 3: Critical modes for Nov 18, 2011, Event No 3.

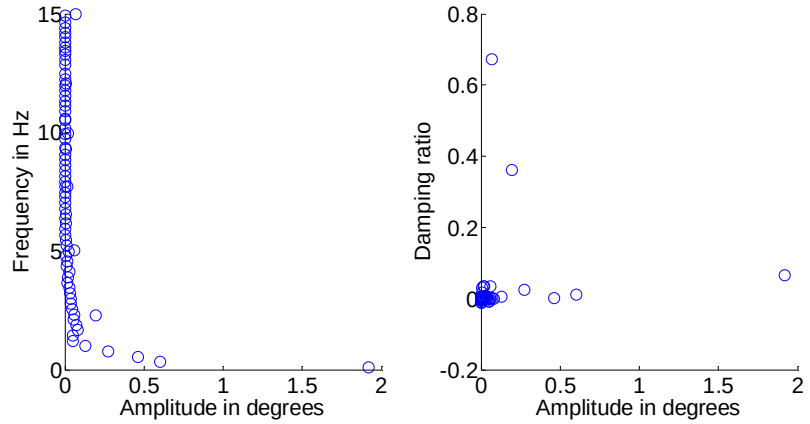

Figure 22: Modal Plot at 05:44:26 GMT for a ten second window.

4.2.2.2 Event No 4; Unit Trip

This event started at 10:57:13 GMT and extended to 10:57:23 GMT

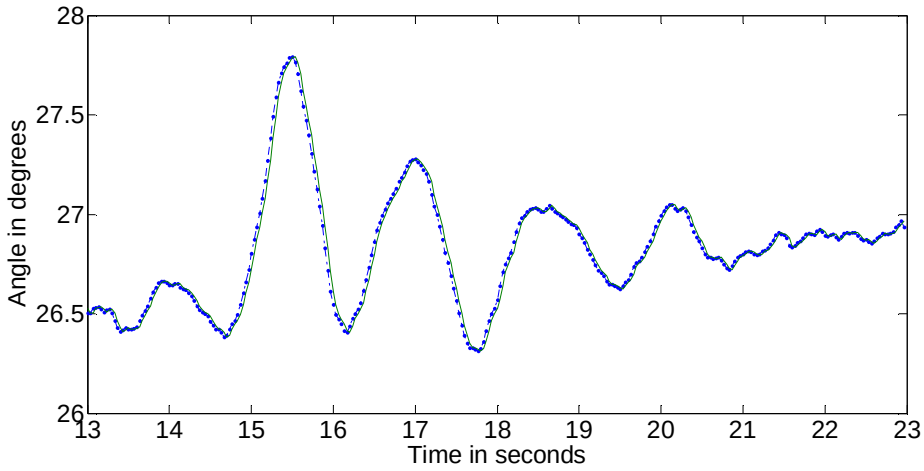

Figure 23: Event No. 4, beginning at 10:57:13 GMT.

For this event we see a damped sinusoid that settled to a new value and thus this is most likely a unit trip. Table 4 gives the dominant mode shape information and Figure 24 shows the modal plot.

FIG NO	HOUR:MIN	START SEC	AMP (DEG)	FREQ (HZ)	DAMPING RATIO	SNR (DB)
23	10:57	13	1.37	0.66	0.086	57.3

Table 4: Critical modes for Nov 18, 2011, Event No 4.

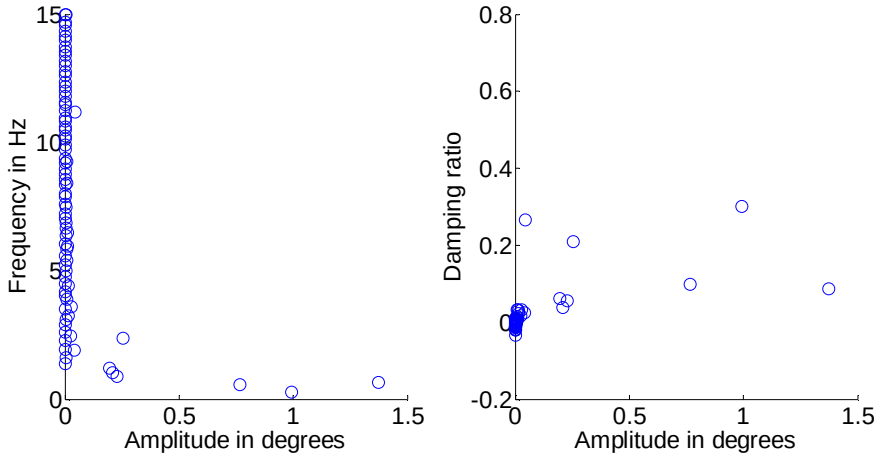

Figure 24: Modal Plot at 10:57:13 GMT for a ten second window.

4.2.2.3 Event No 5

This event started at 19:07:45 GMT and extended to 19:07:55 GMT

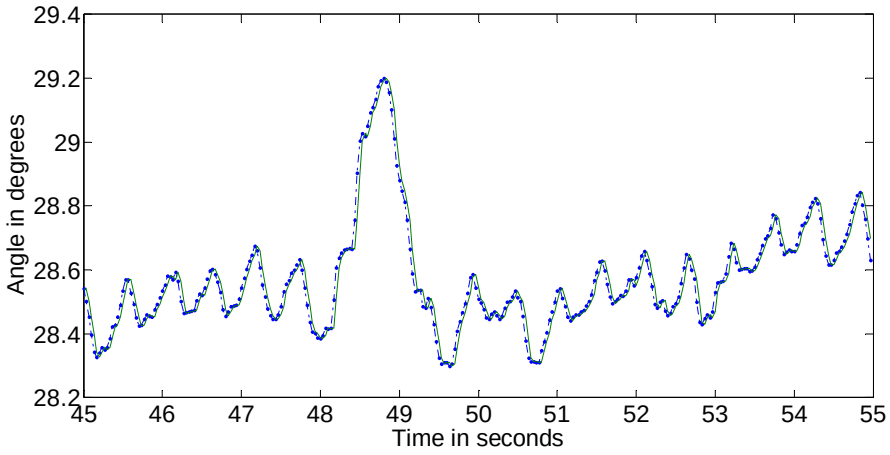

Figure 25: Event No. 5, beginning at 19:07:45 GMT.

The corresponding critical modes are given by Table 5 and modal plot by Figure 26. At this time, we cannot say for certain what caused this event. However since the dominant mode has an amplitude greater than 1º, we can say that it is a significant event.

FIG NO	HOUR:MIN	START SEC	AMP (DEG)	FREQ (HZ)	DAMPING RATIO	SNR (DB)
25	10:57	13	1.37	0.66	0.086	57.3

Table 5: Critical modes for Nov 18, 2011, Event No 5.

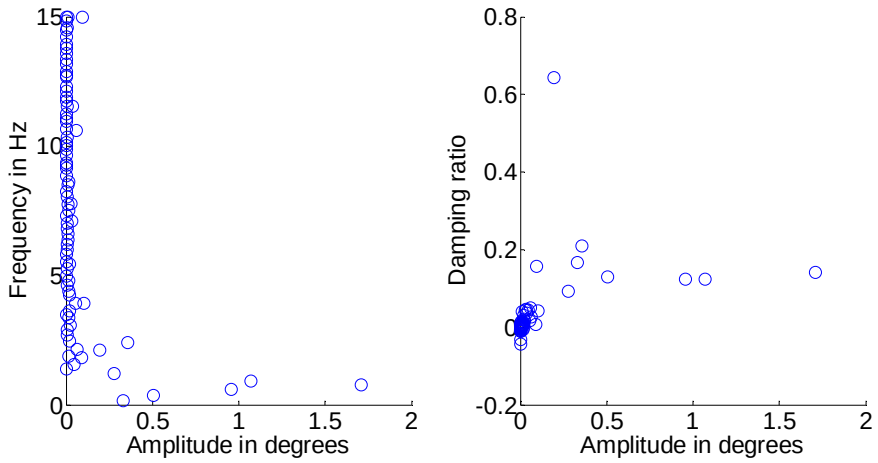

Figure 26: Modal Plot at 19:07:45 GMT for a ten second window.

The above modal plot shows that first three dominant modes have frequencies less than 1 Hz and damping ratio of less than 0.2. So, we can conclude that it is a significant event as the mode frequency lies in the normal 0.5-1.0 Hz.

4.2.3 Result III: 25th Jan 2012

4.2.3.1 Event No 6

During the ten second window starting from 07:52:25 GMT on this day a significant mode with amplitude of 1.8083º was observed for the angle difference between McDonald Observatory and UT Austin. The screening algorithm identified this event when the frequency threshold was raised from 2 Hz to 3 Hz. This event looked similar to the one described above (Event No 5) and is shown below in Figure 27.

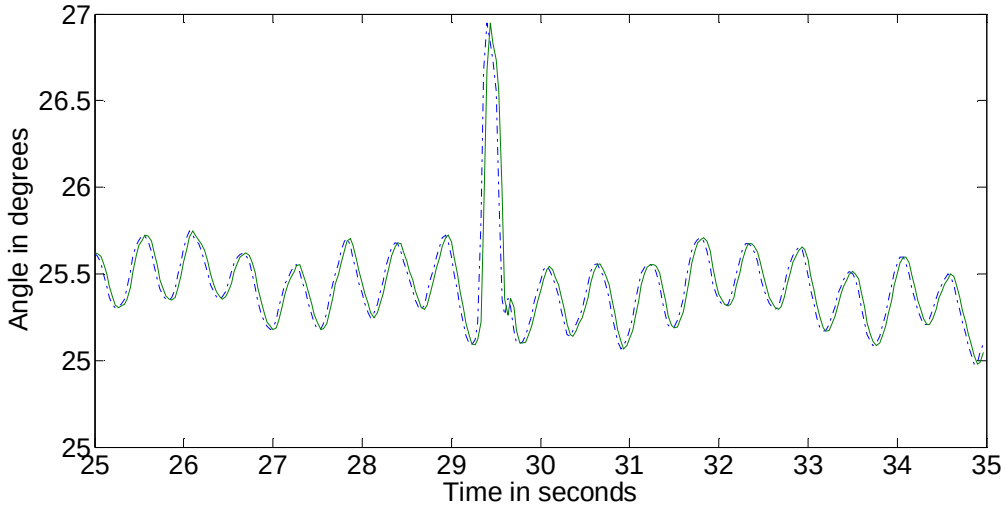

Figure 27: Event No. 6, beginning at 07:52:25 GMT.

The modal plot corresponding to this is shown in Figure 28 and the corresponding modes are given by Table 6. We observe that this mode has a very low damping of 0.0540 and a frequency of 2.51 Hz . Although the exact nature of this event is unknown, it can be concluded based on its modal characteristics that it was a significant event.

FIG NO	HOUR:MIN	START SEC	AMP (DEG)	FREQ (HZ)	DAMPING RATIO	SNR (DB)
27	07:52	25	1.803	2.51	0.05406	42.32

Table 6: Critical modes for Jan 25, 2012, Event No 6.

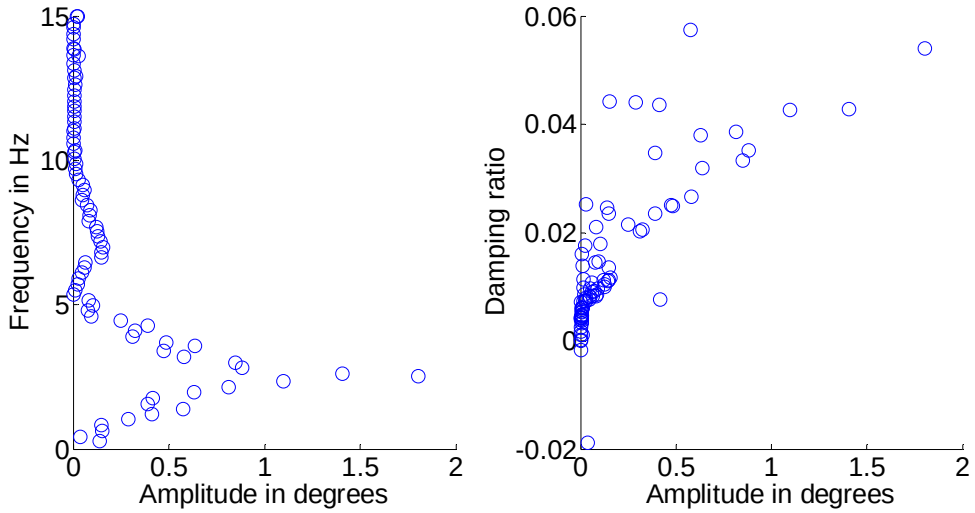

Figure 28: Modal Plot at 07:52:25 GMT for a ten second window.

4.3 RESULTS OF ONE HOUR MODAL PLOTS

The above screening algorithm is helpful in identifying significant power system events and their specific time of occurrence based on the dominant mode amplitude. However, a complete modal plot consisting of all modes within the hour of interest gives more insight into power system behavior. It is informative in the sense that it helps in looking at a particular trend that extended over comparatively a long time or may also be visible throughout the hour which is being analyzed. This is evident from the results below.

4.3.1 Result I: 25th Jan 2012, 10:00:00 GMT

4.3.1.1 Trend No 1: 10:00:00 GMT

During this day an interesting behavior was observed which is captured better if we look at the total one hour modal plot. The angle difference between McDonald Observatory in west Texas and UT Austin is analyzed and the corresponding modal plot

is shown in Figure 29. The following is the value of user defined thresholds (which were described in Chapter 3) used:

- Minimum Amplitude; 0.2º
- Minimum Frequency; 0.01 Hz
- Maximum Frequency; 5Hz
- Maximum Damping; 0.25
- Damping Threshold 1; 0.2
- Damping Threshold 2; 0.1
- Window Length; 10 seconds
- Step size; 5 Seconds

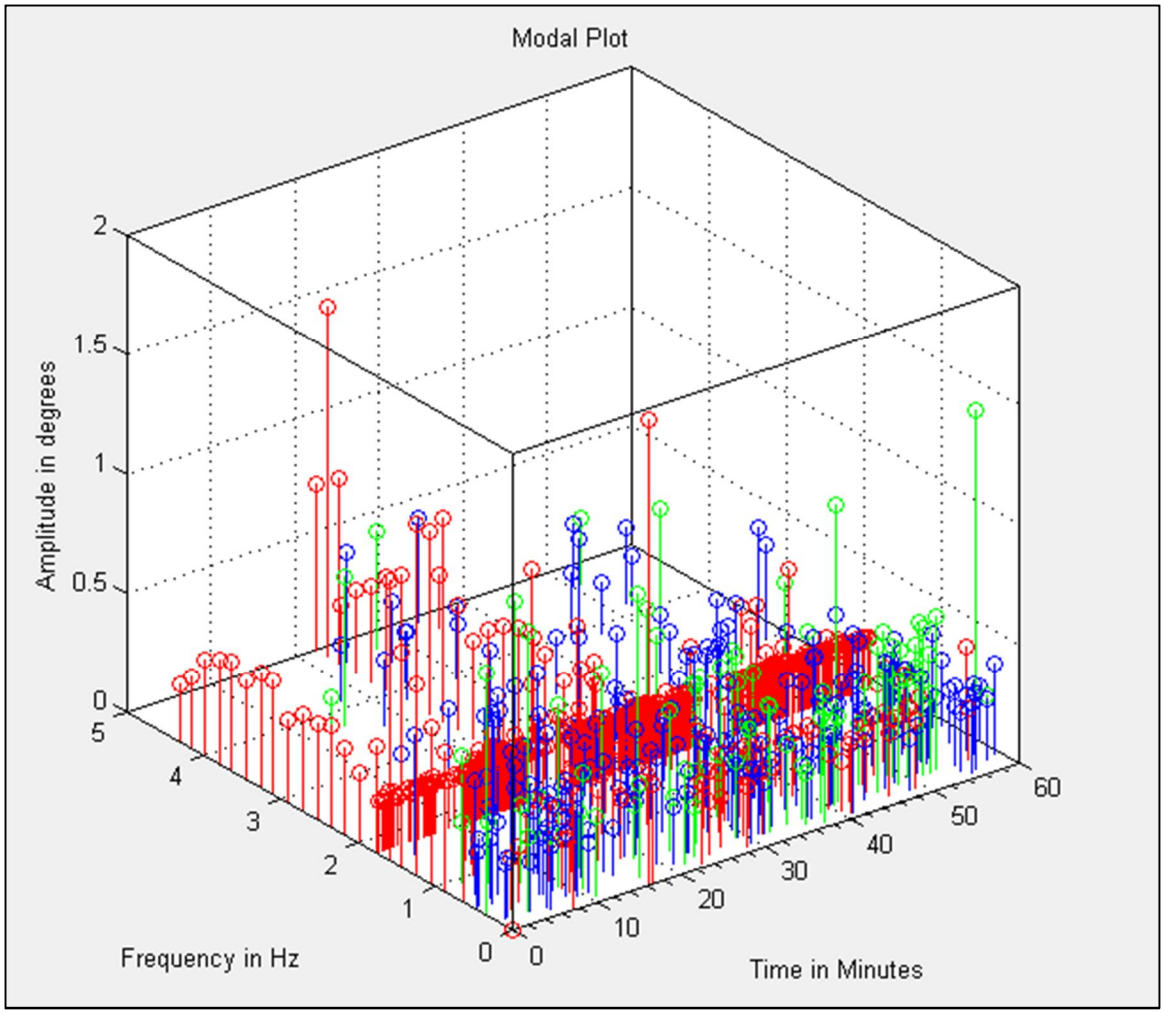

Figure 29: One hour Modal Plot for 10:00:00 GMT, 25th Jan 2012.

As described earlier, damping is shown in the graph by the color of stem points, all modes having a damping ratio between the maximum damping and damping threshold 1 are shown by 'green' color. Modes having a damping ratio value between damping threshold 1 and damping threshold 2 are shown by 'blue' color and modes with a damping ratio less that damping threshold 2 are shown by 'red' color. In this case we observe a trend of consistent occurrence of frequency of 1.8 Hz. Also the modes corresponding to this are in 'red' color implying they all have a damping less than

damping threshold 2 whose value is 0.1 in this case. Thus, these modes are poorly damped.

4.3.1.2 Trend No 2: 07:00:00 GMT

A similar trend was observed on the same day during 07:00:00 GMT; however it was less predominant than the previous one. Here also, we see a cluster of 1.8 Hz modes which are poorly damped. This is shown in Figure 30. The user defined thresholds are the same as used in the previous case.

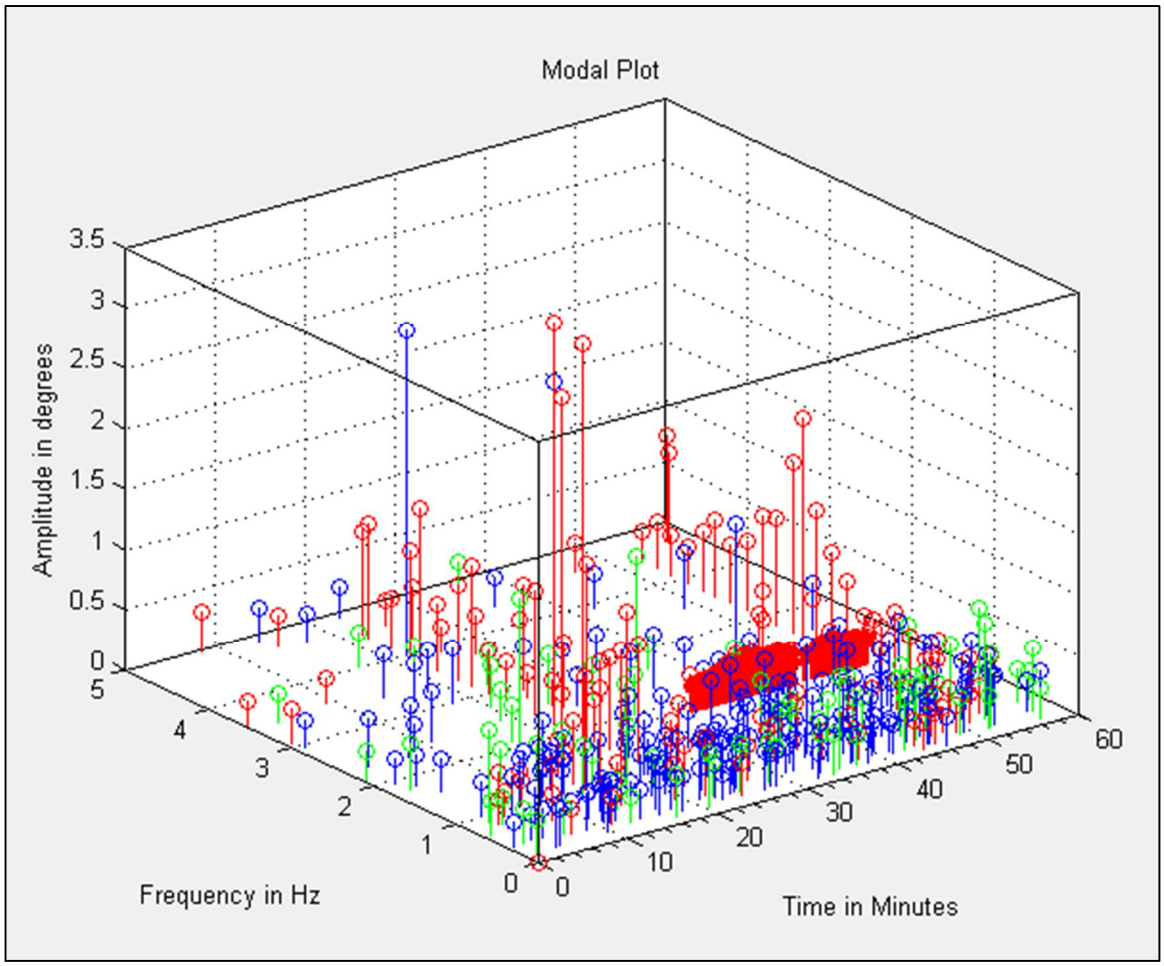

Figure 30: One hour Modal Plot for 07:00:00 GMT, 25th Jan 2012

CONCLUSIONS

This thesis uses modal analysis as a tool to identify power system events. It does so by developing a new screening procedure that uses dominant mode characteristics to capture events. This screening procedure is being used by the Texas Synchrophasor Network to examine PMU data for power system events of interest. It supplements our presently used FFT based screening procedure. More than 2.5 million phase angle measurements per PMU are recorded daily by the Texas Synchrophasor Network. The screening technique is essential for capturing with high confidence the few significant events that happen every day. The results show that the algorithm is successful in identifying events, also since the algorithm is based on modal characteristics it also provides with modal information which gives further insight into the power system behavior. The process is explained in detail and is illustrated with six actual events.

Extending the algorithm to build a user interface provided with a platform to detect certain trends and comment better about the system behavior. This was illustrated with two trends which shows a steady mode frequency of 1.8 Hz.

Looking at the results it can be stated with confidence that through further experience and study of events captured by the Texas Synchrophasor Network within ERCOT we can in future better characterize the many types of events that are observed.

APPENDIX

MATLAB CODE

```
function varargout = Sant_Thesis_GUI(varargin)
% SANT_THESIS_GUI M-file for Sant_Thesis_GUI.fig
%      SANT_THESIS_GUI, by itself, creates a new SANT_THESIS_GUI or raises the
existing
%      singleton*.
%
%      H = SANT_THESIS_GUI returns the handle to a new SANT_THESIS_GUI or the
handle to
%      the existing singleton*.
%
%      SANT_THESIS_GUI('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in SANT_THESIS_GUI.M with the given input
arguments.
%
%      SANT_THESIS_GUI('Property','Value',...) creates a new SANT_THESIS_GUI or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Sant_Thesis_GUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Sant_Thesis_GUI_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Sant_Thesis_GUI

% Last Modified by GUIDE v2.5 15-Apr-2012 17:24:26

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Sant_Thesis_GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @Sant_Thesis_GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before Sant_Thesis_GUI is made visible.
function Sant_Thesis_GUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Sant_Thesis_GUI (see VARARGIN)

% Choose default command line output for Sant_Thesis_GUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


% UIWAIT makes Sant_Thesis_GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = Sant_Thesis_GUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

%This is the code  segment for the ‘Browse Button that loads the .csv file
%from MATLAB folder
% --- Executes on button press in pushbuttonBrowse.
function editLoad_hour_Callback(hObject, eventdata, handles)
% hObject    handle to editLoad_hour (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editLoad_hour as text
%        str2double(get(hObject,'String')) returns contents of editLoad_hour as
a double


% --- Executes during object creation, after setting all properties.
function editLoad_hour_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editLoad_hour (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenuY.
%Callback and CreateFcn for the 'popupmenyY' that loads the location of
```

```
%synchrophasor data to be analyzed
function popupmenuY_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenuY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
guidata(hObject,handles);
% Hints: contents = get(hObject,'String') returns popupmenuY contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from popupmenuY

% --- Executes during object creation, after setting all properties.
function popupmenuY_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenuY (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenuR.
%Callback and CreateFcn for the 'popupmenyR' that loads the reference of
synchrophasor data to be analyzed
function popupmenuR_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenuR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
guidata(hObject,handles);
% Hints: contents = get(hObject,'String') returns popupmenuR contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from popupmenuR

% --- Executes during object creation, after setting all properties.
function popupmenuR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenuR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbuttonR.


function editS_Callback(hObject, eventdata, handles)
% hObject    handle to editS (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of editS as text
%        str2double(get(hObject,'String')) returns contents of editS as a
double
```

```
% --- Executes during object creation, after setting all properties.
function editS_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editS (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editW_Callback(hObject, eventdata, handles)
% hObject    handle to editW (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editW as text
%        str2double(get(hObject,'String')) returns contents of editW as a
double

% --- Executes during object creation, after setting all properties.
function editW_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editW (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function editampMin_Callback(hObject, eventdata, handles)
% hObject    handle to editampMin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
guidata(hObject,handles);
% Hints: get(hObject,'String') returns contents of editampMin as text
%        str2double(get(hObject,'String')) returns contents of editampMin as a
double


% --- Executes during object creation, after setting all properties.
function editampMin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editampMin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function editFreqMin_Callback(hObject, eventdata, handles)
% hObject    handle to editFreqMin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editFreqMin as text
%        str2double(get(hObject,'String')) returns contents of editFreqMin as a
double


% --- Executes during object creation, after setting all properties.
function editFreqMin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editFreqMin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editFreqmax_Callback(hObject, eventdata, handles)
% hObject    handle to editFreqmax (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editFreqmax as text
%        str2double(get(hObject,'String')) returns contents of editFreqmax as a
double


% --- Executes during object creation, after setting all properties.
function editFreqmax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editFreqmax (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editDampMax_Callback(hObject, eventdata, handles)
% hObject    handle to editDampMax (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editDampMax as text
```

```
%        str2double(get(hObject,'String')) returns contents of editDampMax as a
double


% --- Executes during object creation, after setting all properties.
function editDampMax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editDampMax (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editDampThresh2_Callback(hObject, eventdata, handles)
% hObject    handle to editDampThresh2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editDampThresh2 as text
%        str2double(get(hObject,'String')) returns contents of editDampThresh2
as a double


% --- Executes during object creation, after setting all properties.
function editDampThresh2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editDampThresh2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%This is the code segment for the ‘Browse Button’ that loads the ‘.csv’ file
%from MATLAB folder
% --- Executes on button press in pushbuttonBrowse.
function pushbuttonBrowse_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonBrowse (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.filename=uigetfile('*.csv');
guidata(hObject,handles)
setedit1String(handles.editLoad_hour,eventdata, handles)
setPopupmenuString(handles.popupmenuR,eventdata, handles)
setPopupmenuString(handles.popupmenuY,eventdata, handles)

% setedit1String is the function that takes the name of the .’csv’ file
% selected and displays it in the load hour edit button
function setedit1String(hObject, eventdata, handles)
filename=handles.filename;
```

```
set(hObject,'string',filename);

%This is the code segment for the two ‘popupmenus’ that contains the column
names with each column corresponding to a data set from a synchrophasor
location
function setPopupmenuString(hObject,eventdata, handles)
filename =handles.filename;
num=importdata(filename);
colNames=num.textdata(1,:);
set(hObject, 'string',colNames);

function pushbuttonR_Callback(hObject, eventdata, handles)
%tic starts the timer to obatin the total time taken to execute the 'run'
%button that contains the main code for the modal plot and the screening
%algorithm
tic
% q is a variable that is initialized to zero to start a array with first
% row corresponding to q+1 that stores the dominant mode characteristics
% obtained form the screening algorithm, this is the 'Result' array.
q=0;
 % we first obtain the values from all the 'edit button' of the GUI
%savefile contains the name of the file defined by user which will store
%the Result array or the screened dominanat mode characteristics in the
%.xls format
savefile=get(handles.edit_savefile,'String');
% stepsize is defined in the thesis section 3.2.2.7
stepsize=str2num(get(handles.editS,'String'));
% window length is defined in thesis section 3.2.2.6
W=str2num(get(handles.editW,'String'));
% Amplitude thresholds are defined in section 3.2.2.8 and 3.2.2.9
Amp_min=str2num(get(handles.editampMin,'String'));
Amp_screen=str2num(get(handles.editScreen_amp,'String'));
% Frequency thresholds are defined in thesis section 3.2.2.10
Freq_max=str2num(get(handles.editFreqmax,'String'));
Freq_min=str2num(get(handles.editFreqMin,'String'));
%Damping thresholds are defined in thesis section 3.2.2.11
Damp_max=str2num(get(handles.editDampMax,'String'));
Damp_Thresh1=str2num(get(handles.editDampThresh1,'String'));
Damp_Thresh2=str2num(get(handles.editDampThresh2,'String'));
% rcol and ycol if the number referencing to the column that contains the
% reference and loaction synchrophasor data respectively
rcolname = get(handles.popupmenuR,'value');
rcol=rcolname-1
ycolname = get(handles.popupmenuY,'value');
ycol = ycolname - 1
filename = handles.filename;

%num contains the textdata and numericla data imported from the '.csv' file
num=importdata(filename);

%Sheet extracts only the numeric data from num and calculates the final
%value of location data (corresponding to ycol) with respect to reference
%data (corresponding to r col)
Sheet=num.data(:,ycol)-num.data(:,rcol);

%col_W is the coloumns for storing individual window data for the hour i.e
%each modes obatined from a window length W is stored in a new coloumn that
%is updated thorugh the counter col_W
col_W=1;
```

```
%frequency of sampling=f_sampling;

f_sampling=30;

% obtaining the time period of analysis from the 'edit buttons'
% corresponding to them, this is defined in thesis section 3.2.2.5
%Specify the start minute=M_start
M_start=str2num(get(handles.editStartMin,'String'));

%Specify the start second=S_start
S_start=str2num(get(handles.editStartSec,'String'));

%specify the stop minute=M_stop
M_stop=str2num(get(handles.editStopMin,'String'));

%specify the stop second=S_stop
S_stop=str2num(get(handles.editStopSec,'String'));

%below is the code that extracts the data from Sheet corresponding to the
%time period defined above and stores it in a different array Sheet_M
Temp=(M_stop-M_start)*60-S_start+S_stop;
Remainder=rem(Temp,W);
sec_total=Temp+W-Remainder;
Total=sec_total*f_sampling;
add=M_start*f_sampling*60+S_start*f_sampling;

for i=1:1:Total
Sheet_M(i,1)=Sheet((i+add),1);
end

for i=1:1:Total
if (Sheet_M(i,1)>=0)
    Sheet_M(i,1)=Sheet_M(i,1);
else
    Sheet_M(i,1)=360+Sheet_M(i,1);
end
end


%factor denotes the  number of data samples contain in step size; this is
%the number of samples by which sliding window will move forward

factor=stepsize*f_sampling;

% we define a new variable counter that contains the current value of start
% second of the current window W
counter=S_start;

%hold off is necessary to refresh the axis everytime 'run' button is
%pressed and newdata is being analyzed
hold off

% now we start the loop that analyzes the whole of data contained in the
% user defined time period and obtains the modal plot and value of dominant
% mode meeting the screening threshold
for s=1:factor:(Total-W*f_sampling)
% N is the total number of data samples which is sampling frequency
% multiplied by window length, also explained in thesis section 3.1.1.2
```

```
N=W*30;

% below is the loop that extracts the data corresponding to the present
% window length W
for i=1:1:N
    value(i)=Sheet_M(i+(s-1),1);

end

%from here the prony algorithm is started, the same algorithm is performed
%for each of the seven LPM order n

%building x matrix defined by equation 3.4 in thesis, we denote this by Y
%here
% we use the first LPM order value as defined by equation 3.14 in thesis
n=(N/2)-20;

v=n+1;
for i=1:1:(N-n)
    Y(i)=value(v);
    v=v+1;
end

%building X matrix defined by equation 3.5 in thesis we denote it by S here
for i=1:1:(N-n)
    f=i;
    for j=n:-1:1
        S(i,j)=value(f);

        f=f+1;
    end
end

%findinding the a coefficients as defined by equation 3.2 and 3.5 in thesis
c=pinv(S);
a=c*Y';

%constructing polynomial given by equation 3.6
A=[1 -a'];

%finding roots
z=roots(A);

%building the Q matrix that is given by equation 3.10

for i=1:1:N
    for j=1:1:n
        Q(i,j)=(z(j)^(i-1));
    end
end

% Using the above to find the B matrix defined by equation 3.10 and 3.11

e=pinv(Q);
B=e*value';

%reconstructing original signal given by equation 3.12
for k=1:1:N
    y_new(n,k)=0;
```

```
    for i=1:1:n
        y_new(n,k)=y_new(n,k)+(B(i))*(z(i)^k);

    end
end

%calculating signal to noise ratio using equation 3.13
r=norm(value)/norm(value-y_new(n,:));
SNR=20*log10(r);
Signal(n)=SNR;

%sampling time=T
T=1/30;

%calculating damping and frequency given by equation 3.8 and 3.9
d=real(log(z))/T;
f=imag(log(z))/(2*pi*T);
ph=phase(B);
D=(-d)./(sqrt(d.^2+(2*pi*f).^2));

%calculating amplitude given by equation 3.11
amp=2*abs(B);

%making a new array in which storing only the values corresponding to which
%frequency is greater than the minimum threshold frequency provided by user,
starting by setting counter e=1
e=1;
for i=1:1:n
    if(f(i)>Freq_min)
        f_F(e,n)=f(i);
        d_F(e,n)=d(i);
        D_F(e,n)=D(i);
        A_F(e,n)=2*abs(B(i));
        B_F(e,n)=B(i);
        ph_F(e,n)=ph(i);
        z_F(e,n)=z(i);
        e=e+1;
    end

end

%code segment to get the dominant mode characteristics
[larg_amp, ndx]=dsort(A_F(:,n));
% storing each of the dominant mode characteristics in new array at a
% location equal to its LPM order n.
Amp(n)=A_F(ndx(1),n);
Freq(n)=f_F(ndx(1),n);
Damp(n)=D_F(ndx(1),n);

%the above code is repeated for the next six LPM order n
n=(N/2)-15
    v=n+1;
for i=1:1:(N-n)
    Y_two(i)=value(v);
    v=v+1;
end

%building S matrix
for i=1:1:(N-n)
```

```
    f=i;
    for j=n:-1:1
        S_two(i,j)=value(f);


        f=f+1;
    end
end
%finding a matrix

c_two=pinv(S_two);
a_two=c_two*Y_two';
%constructing polynomial

A_two=[1 -a_two'];

%finding roots
z_two=roots(A_two);


%building the Q matrix

for i=1:1:N
    for j=1:1:n
        Q_two(i,j)=(z_two(j)^(i-1));
    end
end

% finding B matrix

e_two=pinv(Q_two);
B_two=e_two*value';

%reconstructing original signal
for k=1:1:N
    y_new(n,k)=0;
    for i=1:1:n
        y_new(n,k)=y_new(n,k)+(B_two(i))*(z_two(i)^k);

    end
end

%calculating signal to noise ratio
r=norm(value)/norm(value-y_new(n,:));
SNR=20*log10(r);

Signal(n)=SNR;

%calculating damping and frequency
d=real(log(z_two))/T;
f=imag(log(z_two))/(2*pi*T);
ph=phase(B_two);
D=(-d)./(sqrt(d.^2+(2*pi*f).^2));
amp=2*abs(B_two);

%making a new array in which storing only the values corresponding to which
%frequency is greater than min frequency threshold, starting by setting counter
e=1
e=1;
```

```
for i=1:1:n
    if(f(i)>Freq_min)
        f_F(e,n)=f(i);
        d_F(e,n)=d(i);
        D_F(e,n)=D(i);
        A_F(e,n)=2*abs(B_two(i));
        B_F(e,n)=B_two(i);
        ph_F(e,n)=ph(i);
        z_F(e,n)=z_two(i);
        e=e+1;
    end

end
[larg_amp, ndx]=dsort(A_F(:,n));
A_F(ndx(1),n)

Amp(n)=A_F(ndx(1),n);
Freq(n)=f_F(ndx(1),n);
Damp(n)=D_F(ndx(1),n);

%the third LPM order n
   n=(N/2)-10;
    v=n+1;
for i=1:1:(N-n)
    Y_three(i)=value(v);
    v=v+1;
end

%building S matrix
for i=1:1:(N-n)
    f=i;
    for j=n:-1:1
        S_three(i,j)=value(f);


        f=f+1;
    end
end
%finding a matrix

c_three=pinv(S_three);
a_three=c_three*Y_three';
%constructing polynomial

A_three=[1 -a_three'];

%finding roots
z_three=roots(A_three);


%building the Q matrix

for i=1:1:N
    for j=1:1:n
        Q_three(i,j)=(z_three(j)^(i-1));
    end
end

% finding B matrix
```

```
e_three=pinv(Q_three);
B_three=e_three*value';

%reconstructing original signal
for k=1:1:N
    y_new(n,k)=0;
    for i=1:1:n
        y_new(n,k)=y_new(n,k)+(B_three(i))*(z_three(i)^k);

    end
end



%calculating signal to noise ratio
r=norm(value)/norm(value-y_new(n,:));
SNR=20*log10(r);
Signal(n)=SNR;



%calculating damping and frequency
d=real(log(z_three))/T;


f=imag(log(z_three))/(2*pi*T);
ph=phase(B_three);
D=(-d)./(sqrt(d.^2+(2*pi*f).^2));
amp=2*abs(B_three);



%making a new array in which storing only the values corresponding to which
%frequency is greater than minimum frequency threshold, starting by setting
counter e=1
e=1;
for i=1:1:n
    if(f(i)>Freq_min)
        f_F(e,n)=f(i);
        d_F(e,n)=d(i);
        D_F(e,n)=D(i);
        A_F(e,n)=2*abs(B_three(i));
        B_F(e,n)=B_three(i);
        ph_F(e,n)=ph(i);
        z_F(e,n)=z_three(i);
        e=e+1;
    end

end
[larg_amp, ndx]=dsort(A_F(:,n));
A_F(ndx(1),n)

Amp(n)=A_F(ndx(1),n);
Freq(n)=f_F(ndx(1),n);
Damp(n)=D_F(ndx(1),n);

% fourth LPM order n
    n=(N/2)-4;
```

```
    v=n+1;
for i=1:1:(N-n)
    Y_four(i)=value(v);
    v=v+1;
end

%building S matrix
for i=1:1:(N-n)
    f=i;
    for j=n:-1:1
        S_four(i,j)=value(f);


        f=f+1;
    end
end
%finding a matrix
%Sa=Y

c_four=pinv(S_four);
a_four=c_four*Y_four';
%constructing polynomial

A_four=[1 -a_four'];

%finding roots
z_four=roots(A_four);


%building the Q matrix

for i=1:1:N
    for j=1:1:n
        Q_four(i,j)=(z_four(j)^(i-1));
    end
end

% finding B matrix

e_four=pinv(Q_four);
B_four=e_four*value';

%reconstructing original signal
for k=1:1:N
    y_new(n,k)=0;
    for i=1:1:n
        y_new(n,k)=y_new(n,k)+(B_four(i))*(z_four(i)^k);

    end
end


%calculating signal to noise ratio
r=norm(value)/norm(value-y_new(n,:));
SNR=20*log10(r);
Signal(n)=SNR;
```

```
%calculating damping and frequency
d=real(log(z_four))/T;


f=imag(log(z_four))/(2*pi*T);
ph=phase(B_four);
D=(-d)./(sqrt(d.^2+(2*pi*f).^2));
amp=2*abs(B_four);



%making a new array in which storing only the values corresponding to which
%frequency is greater than mimimum frequency threshold, starting by setting
counter e=1
e=1;
for i=1:1:n
    if(f(i)>Freq_min)
        f_F(e,n)=f(i);
        d_F(e,n)=d(i);
        D_F(e,n)=D(i);
        A_F(e,n)=2*abs(B_four(i));
        B_F(e,n)=B_four(i);
        ph_F(e,n)=ph(i);
        z_F(e,n)=z_four(i);
        e=e+1;
    end

end
[larg_amp, ndx]=dsort(A_F(:,n));
A_F(ndx(1),n)

Amp(n)=A_F(ndx(1),n);
Freq(n)=f_F(ndx(1),n);
Damp(n)=D_F(ndx(1),n);


% fifth LPM order n
    n=(N/2)+6;
    v=n+1;
for i=1:1:(N-n)
    Y_five(i)=value(v);
    v=v+1;
end

%building S matrix
for i=1:1:(N-n)
    f=i;
    for j=n:-1:1
        S_five(i,j)=value(f);


        f=f+1;
    end
end
%finding a matrix
%Sa=Y

c_five=pinv(S_five);
a_five=c_five*Y_five';
```

```
%constructing polynomial

A_five=[1 -a_five'];

%finding roots
z_five=roots(A_five);


%building the Q matrix

for i=1:1:N
    for j=1:1:n
        Q_five(i,j)=(z_five(j)^(i-1));
    end
end

% finding B matrix

e_five=pinv(Q_five);
B_five=e_five*value';

%reconstructing original signal
for k=1:1:N
    y_new(n,k)=0;
    for i=1:1:n
        y_new(n,k)=y_new(n,k)+(B_five(i))*(z_five(i)^k);

    end
end


%calculating signal to noise ratio
r=norm(value)/norm(value-y_new(n,:));
SNR=20*log10(r);
Signal(n)=SNR;

%calculating damping and frequency
d=real(log(z_five))/T;


f=imag(log(z_five))/(2*pi*T);
ph=phase(B_five);
D=(-d)./(sqrt(d.^2+(2*pi*f).^2));
amp=2*abs(B_five);



%making a new array in which storing only the values corresponding to which
%frequency is greater than zero.2, starting by setting counter e=1
e=1;
for i=1:1:n
    if(f(i)>Freq_min)
        f_F(e,n)=f(i);
        d_F(e,n)=d(i);
        D_F(e,n)=D(i);
        A_F(e,n)=2*abs(B_five(i));
        B_F(e,n)=B_five(i);
        ph_F(e,n)=ph(i);
        z_F(e,n)=z_five(i);
```

```
        e=e+1;
    end

end
[larg_amp, ndx]=dsort(A_F(:,n));
A_F(ndx(1),n)
Amp(n)=A_F(ndx(1),n);
Freq(n)=f_F(ndx(1),n);
Damp(n)=D_F(ndx(1),n);

%sixth LPM order n
n=(N/2)+10;
    v=n+1;
for i=1:1:(N-n)
    Y_six(i)=value(v);
    v=v+1;
end

%building S matrix
for i=1:1:(N-n)
    f=i;
    for j=n:-1:1
        S_six(i,j)=value(f);


        f=f+1;
    end
end
%finding a matrix
%Sa=Y

c_six=pinv(S_six);
a_six=c_six*Y_six';
%constructing polynomial

A_six=[1 -a_six'];

%finding roots
z_six=roots(A_six);


%building the Q matrix

for i=1:1:N
    for j=1:1:n
        Q_six(i,j)=(z_six(j)^(i-1));
    end
end

% finding B matrix

e_six=pinv(Q_six);
B_six=e_six*value';

%reconstructing original signal
for k=1:1:N
    y_new(n,k)=0;
    for i=1:1:n
        y_new(n,k)=y_new(n,k)+(B_six(i))*(z_six(i)^k);
```

```
    end
end


%calculating signal to noise ratio
r=norm(value)/norm(value-y_new(n,:));
SNR=20*log10(r);
Signal(n)=SNR;

%calculating damping and frequency
d=real(log(z_six))/T;


f=imag(log(z_six))/(2*pi*T);
ph=phase(B_six);
D=(-d)./(sqrt(d.^2+(2*pi*f).^2));
amp=2*abs(B_six);



%making a new array in which storing only the values corresponding to which
%frequency is greater than minimum frequency threshold, starting by setting
counter e=1
e=1;
for i=1:1:n
    if(f(i)>Freq_min)
        f_F(e,n)=f(i);
        d_F(e,n)=d(i);
        D_F(e,n)=D(i);
        A_F(e,n)=2*abs(B_six(i));
        B_F(e,n)=B_six(i);
        ph_F(e,n)=ph(i);
        z_F(e,n)=z_six(i);
        e=e+1;
    end

end
[larg_amp, ndx]=dsort(A_F(:,n));
A_F(ndx(1),n)
Amp(n)=A_F(ndx(1),n);
Freq(n)=f_F(ndx(1),n);
Damp(n)=D_F(ndx(1),n);

%startof seven LPM order
n=(N/2)+17;
    v=n+1;
for i=1:1:(N-n)
    Y_seven(i)=value(v);
    v=v+1;
end

%building S matrix
for i=1:1:(N-n)
    f=i;
    for j=n:-1:1
        S_seven(i,j)=value(f);
```

```
        f=f+1;
    end
end
%finding a matrix
%Sa=Y

c_seven=pinv(S_seven);
a_seven=c_seven*Y_seven';
%constructing polynomial

A_seven=[1 -a_seven'];

%finding roots
z_seven=roots(A_seven);


%building the Q matrix

for i=1:1:N
    for j=1:1:n
        Q_seven(i,j)=(z_seven(j)^(i-1));
    end
end

% finding B matrix

e_seven=pinv(Q_seven);
B_seven=e_seven*value';

%reconstructing original signal
for k=1:1:N
    y_new(n,k)=0;
    for i=1:1:n
        y_new(n,k)=y_new(n,k)+(B_seven(i))*(z_seven(i)^k);

    end
end


%calculating signal to noise ratio
r=norm(value)/norm(value-y_new(n,:));
SNR=20*log10(r);
Signal(n)=SNR;

%calculating damping and frequency
d=real(log(z_seven))/T;


f=imag(log(z_seven))/(2*pi*T);
ph=phase(B_seven);
D=(-d)./(sqrt(d.^2+(2*pi*f).^2));
amp=2*abs(B_seven);
e=1;
for i=1:1:n
    if(f(i)>Freq_min)
        f_F(e,n)=f(i);
        d_F(e,n)=d(i);
        D_F(e,n)=D(i);
        A_F(e,n)=2*abs(B_seven(i));
```

```
        B_F(e,n)=B_seven(i);
        ph_F(e,n)=ph(i);
        z_F(e,n)=z_seven(i);
        e=e+1;
    end

end
[larg_amp, ndx]=dsort(A_F(:,n));
A_F(ndx(1),n)
Amp(n)=A_F(ndx(1),n);
Freq(n)=f_F(ndx(1),n);
Damp(n)=D_F(ndx(1),n);
%end of seven


%we obtain the value of LPM order n that gives the maximum SNR
[maxsingnal, sdx]=dsort(Signal);

%calculate the total number of rows in the Matrix A_F that contains all the
%modes corresonding to each of the LPM order n, the number of rows gives
%the total number of modes obatined
e=1;
[row,col]=size(A_F);

%to find the corresponding time matrix for plotting
T(1,col_W)=(s-1)/(30*60);
for i=2:1:row
    T(i,col_W)=T(i-1,col_W)+W/(row*60);
end

% we check each of the modes to meet the amplitude, frequency and damping
% thresholds

for i=1:1:row

if (A_F(i,sdx(1))>=Amp_min && f_F(i,sdx(1))<=Freq_max &&
D_F(i,sdx(1))<=Damp_max)
A_F_b(e,col_W)=A_F(i,sdx(1));
    f_F_b(e,col_W)=f_F(i,sdx(1));
    D_F_b(e,col_W)=D_F(i,sdx(1));
    T_F_b(e,col_W)=T(i,col_W);
    e=e+1;

end

end

% below is the code segment to plot the 3D graph by dividing it into three
% parts corresponding to the damping thresholds defined and plotting the
% points with the corresponding colors defined in section 3.2.2.11
e1=1;
e2=1;
e3=1;
if(e>1)
    [r_F c_F]=size(A_F_b);
    for i=1:1:r_F
    if(D_F_b(i,col_W)<=Damp_max && D_F_b(i,col_W)>Damp_Thresh1)
        D_F1(e1,col_W)=D_F_b(i,col_W);
        f_F1(e1,col_W)=f_F_b(i,col_W);
```

```
        A_F1(e1,col_W)=A_F_b(i,col_W);
        T_F1(e1,col_W)=T_F_b(i,col_W);

        e1=e1+1;
    end
    if(D_F_b(i,col_W)<=Damp_Thresh1 && D_F_b(i,col_W)>Damp_Thresh2)
        D_F2(e2,col_W)=D_F_b(i,col_W);
        f_F2(e2,col_W)=f_F_b(i,col_W);
        A_F2(e2,col_W)=A_F_b(i,col_W);
        T_F2(e2,col_W)=T_F_b(i,col_W);
        e2=e2+1;
    end
     if(D_F_b(i,col_W)<=Damp_Thresh2)
        D_F3(e3,col_W)=D_F_b(i,col_W);
        f_F3(e3,col_W)=f_F_b(i,col_W);
        A_F3(e3,col_W)=A_F_b(i,col_W);
        T_F3(e3,col_W)=T_F_b(i,col_W);
        e3=e3+1;
     end

end
end


if(e1>1)
stem3(handles.ModalPlot,T_F1,f_F1,A_F1,'g');
title('Modal Plot');
xlabel('Time in Minutes');
ylabel('Frequency in Hz');
zlabel('Amplitude in degrees');
set(handles.ModalPlot,'XMinorTick','on')
hold on
end

if(e2>1)
stem3(handles.ModalPlot,T_F2,f_F2,A_F2,'b');
title('Modal Plot');
xlabel('Time in Minutes');
ylabel('Frequency in Hz');
zlabel('Amplitude in degrees');
set(handles.ModalPlot,'XMinorTick','on')
hold on
end

if(e3>1)
stem3(handles.ModalPlot,T_F3,f_F3,A_F3,'r');
title('Modal Plot');
xlabel('Time in Minutes');
ylabel('Frequency in Hz');
zlabel('Amplitude in degrees');
set(handles.ModalPlot,'XMinorTick','on')
hold on
end

%Screening Algorithm, this is the screening procedure defined in section
%3.1.2.2
if(Amp(N/2+17)>Amp_screen&&Amp(N/2-
4)>Amp_screen&&Amp(N/2+10)>Amp_screen&&Amp(N/2+6)>Amp_screen&&Amp(N/2-
15)>Amp_screen&&Amp(N/2-10)>Amp_screen&&Amp(N/2-20)>Amp_screen)
```

```
    [maxsingnal, mdx]=dsort(Signal);
    if(Freq(mdx(1))<Freq_max&&Damp(mdx(1))<Damp_max)
        %first column corresponds to the minute
        Result(q+1,1)=M_start;
        %second column corresponds to the start second of window
        Result(q+1,2)=counter;
        %third column corresponds to the end second of window
        Result(q+1,3)=Result(q+1,2)+W;
        %fourth column contains the dominant mode amplitude
        Result(q+1,4)=Amp(mdx(1));
        %fifth column contains the dominant mode frequency
        Result(q+1,5)=Freq(mdx(1));
        %sixth column contains the dominant mode damping
        Result(q+1,6)=Damp(mdx(1));
         %seventh column contains the SNR of this window lenth W
        Result(q+1,7)=Signal(mdx(1));
        %last column corresponds to the LPM order which gave the maximum
        %SNR and to which all the above values correspond
        Result(q+1,8)=mdx(1);

    q=q+1;
    end
end
    counter=counter+stepsize;


   %code for updating the minute
    if(counter==60)
        counter=0;
        M_start=M_start+1
    else
        counter=counter;
    end

    col_W=col_W+1;

end

%code for updating the slider bar for the zaxis
axis_z=ZLim;
Zmax=axis_z(1,2);
slidemax=get(handles.slider_Amp_zaxis,'Max');
set(handles.slider_Amp_zaxis,'Max',Zmax);
set(handles.slider_Amp_zaxis,'Value',Zmax);

%storing screening results in a .xls file or otherwise giving a message if no
%results were obtained
if(q>0)
   header = {'Min' 'Start sec' 'Stop sec' 'Amp' 'Freq' 'Damp' 'SNR' 'n'};
   xlswrite(savefile, header, 'sheetname') % by defualt starts from A1
xlswrite(savefile, Result, 'sheetname','A2') % array under the header.

else
    msgbox('None of the dominant mode meets the screening criterion')
end
%t stores the time taken to run this function
t=toc;

function editDampThresh1_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to editDampThresh1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editDampThresh1 as text
%        str2double(get(hObject,'String')) returns contents of editDampThresh1
as a double


% --- Executes during object creation, after setting all properties.
function editDampThresh1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editDampThresh1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editStartMin_Callback(hObject, eventdata, handles)
% hObject    handle to editStartMin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editStartMin as text
%        str2double(get(hObject,'String')) returns contents of editStartMin as
a double


% --- Executes during object creation, after setting all properties.
function editStartMin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editStartMin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editStartSec_Callback(hObject, eventdata, handles)
% hObject    handle to editStartSec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editStartSec as text
%        str2double(get(hObject,'String')) returns contents of editStartSec as
a double
```

```
% --- Executes during object creation, after setting all properties.
function editStartSec_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editStartSec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editStopMin_Callback(hObject, eventdata, handles)
% hObject    handle to editStopMin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editStopMin as text
%        str2double(get(hObject,'String')) returns contents of editStopMin as a
double


% --- Executes during object creation, after setting all properties.
function editStopMin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editStopMin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editStopSec_Callback(hObject, eventdata, handles)
% hObject    handle to editStopSec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editStopSec as text
%        str2double(get(hObject,'String')) returns contents of editStopSec as a
double


% --- Executes during object creation, after setting all properties.
function editStopSec_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editStopSec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function editScreen_amp_Callback(hObject, eventdata, handles)
% hObject    handle to editScreen_amp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editScreen_amp as text
%        str2double(get(hObject,'String')) returns contents of editScreen_amp
as a double


% --- Executes during object creation, after setting all properties.
function editScreen_amp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editScreen_amp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on slider movement.
%below is the code for the z axis slider
function slider_Amp_zaxis_Callback(hObject, eventdata, handles)
current_slider=get(handles.slider_Amp_zaxis,'Value');
if(current_slider<0.001)
     current_slider=0.001;
end
 set(handles.ModalPlot,'zlim',[0 current_slider]);
% --- Executes during object creation, after setting all properties.
function slider_Amp_zaxis_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider_Amp_zaxis (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
%Zm=get(handles.ModalPlot,'ZLim')
%Zmax=Zm(1,2)
%set(hObject,'Max', Zmax);
%set(hObject,'Max', Zmax);
%set(hObject,'Value', Zmax);
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end



function edit_savefile_Callback(hObject, eventdata, handles)
% hObject    handle to edit_savefile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_savefile as text
%        str2double(get(hObject,'String')) returns contents of edit_savefile as
a double


% --- Executes during object creation, after setting all properties.
function edit_savefile_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_savefile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

REFERENCES

1. W. Mack Grady, David Costello, "*Implementation and Application of an Independent Texas Synchrophasor Network*", 2010 3rd Annual Conference for Protective Relay Engineers, TAMU, College Station, Tx
2. Texas Synchrophasor Network, *www.ece.utexas.edu/~grady*
3. J.F. Hauer, C.J. Demeure, L.L. Scharf, *"Initial Results in prony Analysis of Power System Response Signals"*, IEEE Transactions on Power Systems, Vol5,No. 1, February 1990.
4. Yanfeng Gong, Armando Guzman, *"Synchrophasor- Based Online Modal Analysis to Mitigate Power System Interarea Oscillation"*, *Schweitzer Engineering Laboratories, Inc.*
5. Aprajita Sant, W. Mack Grady, Surya Santoso, Jamie Ramos, "*A Screening Procedure to detect significant Power System Events recorded by the Texas Synchrophasor Network*", accepted for presentation at the 2012 IEEE Power and Energy Society General Meeting, San Diego, CA.
6. Paulo F. Ribeiro, *"Time-Varying Waveform Distortions in Power Systems"*,Wiley 2009, p 320
7. Zhaoyang Dong, Pei Zhang "*Emerging Techniques in Power System*"; Springer 2010.
8. Phadke A.G., Thorp, J. "*Synchronized Phasor Measurements and their Applications*"; Springer 2010.
9. Glover J. Duncan, Sharma M.S., Overbye Thomas; "*Power System Analysis and Design*"; Thomson learning 2008.
10. Messina A.R. "*Inter-area oscillations in power system*", Springer 2009,6
11. Saeed V Vaseghi, "*Advanced Digital Signal processing and Noise Reduction*", Wiley 2008,p 227.
12. Kunder P., "*Power System Stability and Control*", McGraw-Hill.